COALITION STRUCTURE GENERATION IN CHARACTERISTIC FUNCTION GAMES

By

Travis Service

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

May, 2012

Nashville, Tennessee

Approved:

Dr. Julie A. Adams

Dr. Gautam Biswas

Dr. Jerry S. Spinrad

Dr. Paul H. Edelman

Dr. Vincent Conitzer

Acknowledgements

I would like to first thank my advisor Dr. Julie Adams for her guidance and support of my research as well as the effort and hours she expended reading and correcting drafts of my work. I would also like to thank all of my committee members: Dr. Gautam Biswas, Dr. Jerry Spinrad, Dr. Paul Edelman, and Dr. Vincent Conitzer for helping me pursue my Ph.D. degree. My dissertation research has been supported by an ONR DEPSCOR Award # N000140911161.

I would like to thank Garrett Linn for allowing me to bounce ideas off of him. I would in general also like to thank all of my friends and family.

In particular, I would like to thank my fiancee, Robyn, for being supportive throughout the past four years.

Table of Contents

List of Tables

List of Figures

List of Algorithms

CHAPTER I

Introduction

Cooperation among self-interested agents offers the potential to realize goals unachievable by any agent alone. However, determining the optimal manner in which agents should cooperate is a computationally difficult problem. As an example, imagine that you and $n-1$ of your closest friends are budding entrepreneurs. In an effort to determine how successful your business endeavors will be, each of you sit down and calculate an estimate of your yearly earnings. Now imagine that you and one of your friends have particularly complimentary skills. Say, for example, that on your own you can turn a yearly profit of $\$50,000$ and your friend can earn $\$45,000$. However, by cooperating, given your complimentary skill sets, you could together earn $\$120,000$ yearly. Clearly, this cooperation is desirable for both parties. More generally, assume that k of the individuals working together can earn a larger yearly profit than what the collective earnings of those k individuals will be working alone. Again, in such a situation, cooperation among the k individuals is preferred by all individuals involved.

Two questions naturally arise out of such situations: *"Which groups of agents should cooperate in order to maximize their earnings?"* and *"How should the resulting earnings be split between the agents?"* Moreover these two questions can be related. A group of self-interested agents will only choose to cooperate if the result of cooperating nets them a non-negative increase in utility.

Cooperative game theory has provided a number of different answers to the question of how individuals should distribute their collective earnings. Typically such game theoretic solution concepts focus on allocating the collective earnings in ways that are either fair or stable. Fair payoff allocations give each agent an amount that satisfies some intuitively natural properties. Stable payoff allocations give each agent a sufficient amount in order to

disallow profitable deviations, thus making the coalition or team that the agent belongs to stable. Unfortunately perfect stability is not always possible and even when it is, the stable payoff allocations may not always be the most fair.

Much work in multi-agent coalition and team formation has sidestepped the issue of payoff allocations by assuming the agents are group rational. Assuming group rationality tends to simplify the coalition formation process, as agents are no longer interested in their own profits. When an agent's only goal is to maximize the group's profit, the stability of the resulting coalition is no longer a potential problem. Even with group rational agents, the coalition structure generation problem is computationally difficult due to the exponentially large number of possible coalitions that can form.

This dissertation presents a number of approximation algorithms for coalition structure generation. The presented algorithms improve upon the existing state-of-the-art in coalition structure generation by guaranteeing solutions along with constant factor lower bounds on the quality of the solutions in less than the time required to solve the problem exactly. This dissertation is also the first to present approximation algorithms for coalition structure generation tailored to monotonic domains. This dissertation also considers situations in which some agents may be identical to one another. It is shown that an optimal coalition structure can be found much quicker in situations with bounded agent types.

Optimally coordinating multiple agents becomes even more challenging when the agents are self-interested. From the point of view of the system designer, social welfare maximizing solutions (i.e., those solutions that provide the greatest amount of benefit to the system as a whole) are still desirable. Unfortunately, self-interested agents are not guaranteed to converge to social welfare maximizing (or even pareto optimal) states, as such states are not necessarily stable. A well known example of self-interest resulting in poor system performance is Braess' paradox (Braess, 1968; Braess et al., 2005)[1]. Surprisingly, the rela-

[1]Braess' paradox (Braess, 1968; Braess et al., 2005) states that there are situations where adding extra capacity to a network can reduce the network's overall performance, when individual agents act selfishly. As a concrete example of Braess' paradox consider adding an extra road to a transportation network. There are situations where this addition can result in an increase in travel time for all agents involved, even though the

tionship between optimality and stability in cooperative games has received little attention. This dissertation studies the stability-optimality relationship and demonstrates that both optimal and suboptimal coalition structures can be maximally stable. The stability-optimality relationship restricted to certain natural classes of coalitional games is also studied. It is demonstrated that in some classes only optimal coalition structures can be stable. In other classes it is possible for suboptimal coalition structures to be more stable than all optimal coalition structures.

I.1 Dissertation Roadmap

The remainder of this dissertation is organized as follows. Chapter II presents a number of definitions used throughout this dissertation and formalizes the coalition structure generation problem. Chapter III describes related prior research on the coalition structure generation problem.

Chapter IV presents several approximation algorithms for coalition structure generation in both arbitrary coalitional games as well as monotonic games. Chapter IV also presents the first randomized algorithm for approximation coalition structure generation. Chapter V presents an empirical comparison of the approximation algorithms presented in Chapter IV with existing coalition structure generation algorithms.

In Chapter VI, coalitional games with a bounded number of agent types are considered. It is shown that if there are a bounded number of agent types and the type of each agent is known, then an optimal coalition structure can be computed in polynomial time. The presented algorithm for coalition structure generation in games with a bounded number of agent types was independently discovered by Aziz and Keijzer (Aziz and de Keijzer, 2011). Chapter VI also considers coalition structure generation in a special class of graph games with bounded number of agent types.

Chapter VII considers the relationship between optimality of coalition structures and their stability. Several examples illustrate that, in general, optimal coalition structures are

agents can simply ignore the new route and not incur an increase in travel time.

not necessarily the most stable. Consideration is then turned to compactly representable games. It is shown that of the several compact coalitional games defined in Chapter II, only optimal coalition structures can be maximally stable. Two examples of a compactly representable classes of games where non-optimal coalition structures can be the only maximally stable coalition structures are presented.

This dissertation concludes in Chapter VIII with a summary of the contributions and a presentation of the potential future work.

CHAPTER II

General Definitions

This chapter describes the model of multi-agent cooperation employed in this dissertation and provides preliminary definitions. Some definitions that are relevant to a specific chapter of this dissertation are not presented in this chapter. Instead, such definitions are provided in the specific chapters to which they are relevant.

II.1 Coalitional Games

Given a set of n agents, N, a *coalitional game*, or simply a *game*, is defined on N by a function $v : 2^N \rightarrow \mathbb{R}$, where 2^N is the powerset of N (Jianhua, 1988; Rapoport, 1970; Shoham and Leyton-Brown, 2009). v assigns to each coalition, $C \subseteq N$, of agents a value indicating the joint utility those agents will receive if they form a coalition. v is referred to as the *characteristic function* of the game and is assumed to assign each coalition a non-negative utility.

Definition 1 (coalitional game)**.** *A* coalitional game *is a tuple* (N, v) *where* N *is a set of* n *agents and* $v : 2^N \rightarrow \mathbb{R}$ *is a function assigning each subset of* N *a utility. It is required that* $v(\emptyset) = 0$.

Unless explicitly stated, it is assumed that v maps coalitions of agents to non-negative real numbers. This assumption will be employed throughout the majority of this dissertation.

The coalition N is referred to as the *grand coalition*. Coalitions consisting of a single agent (e.g., $\{i\}$) are referred to as *singleton coalitions*.

The primary question this dissertation considers is which coalitions of agents will form. Each agent may be a member of only one coalition at a time. That is, only non-overlapping coalitions are allowed; however, a growing body of work has considered situations in which

agents may belong to multiple coalitions (Chalkiadakis et al., 2008; Shehory and Kraus, 1998).

A coalition structure specifies a set of coalitions for the N agents, with each agent in exactly one coalition (Sandholm et al., 1999; Shoham and Leyton-Brown, 2009).

Definition 2 (coalition structure)**.** *A* coalition structure *over N, CS, is a partitioning of the agents. Each agent in N is assigned to a single coalition in CS.* $\mathscr{P}$ *is the set of all coalition structures over N.*

v is extended to sets of coalitions as follows. For $S = \{C_1, \cdots, C_k\}$:

$$v(S) = \sum_{C \in S} v(C).$$

$v(S)$ is the social welfare or the value of the set of coalitions S. In particular, for $CS \in \mathscr{P}$

$$v(CS) = \sum_{C \in CS} v(C)$$

is the social welfare of the coalition structure CS. When agents are group rational, it is the sole goal of the agents to maximize the social welfare of the system, that is to find the coalition structure CS^* that maximizes $v(CS^*)$. However, when agents are self-interested, this goal may no longer hold.

The coalition structure generation problem is to identify a coalition structure of maximum social welfare (Sandholm et al., 1999; Rahwan et al., 2009b).

Definition 3 (coalition structure generation problem)**.** *The* coalition structure generation problem *is to find the coalition structure* $CS^* \in \mathscr{P}$ *that maximizes:* $v(CS^*) = \sum_{C \in CS^*} v(C)$. *That is, the coalition structure generation problem is to find:*

$$CS^* \in \underset{CS \in \mathscr{P}}{\operatorname{argmax}}(v(CS)).$$

It is often desirable to measure an agent's contribution to a coalition (i.e., the amount of utility that an agent brings to the coalition). An agent's marginal contribution is one such measure (Jianhua, 1988; Rapoport, 1970; Shoham and Leyton-Brown, 2009).

Definition 4 (marginal contribution)**.** *Agent i's* marginal contribution *to a coalition* $C \subseteq N \setminus \{i\}$ *is the value* $v(C \cup \{i\}) - v(C)$.

An agent i's marginal contribution to a coalition C, a coalition that the agent is not already a member of, is the gain in utility that C will receive if i joins the coalition.

There are several classes of coalitional games characterized by the properties of their characteristic functions. A few that are relevant are now introduced.

Definition 5 (monotonic)**.** *A characteristic function* v *is* monotonic *if for all* $C, S \subseteq N$ *such that* $C \subseteq S$, $v(C) \leq v(S)$.

Intuitively, a characteristic function is monotonic if adding an agent to a coalition can never harm the coalition (Jianhua, 1988; Rapoport, 1970; Shoham and Leyton-Brown, 2009). Monotonicity is a natural condition since, at the very least, additional agents that join a coalition can remain inactive and the original set of agents can earn the same amount of utility as the coalition without the inactive new agents.

Definition 6 (superadditive)**.** *A characteristic function* v *is* superadditive *if for all* $C, S \subseteq N$ *with* $C \cap S = \emptyset$, $v(C) + v(S) \leq v(C \cup S)$.

Superadditivity is a common assumption on the basis that if two disjoint coalitions C_1 and C_2 merge into $C_1 \cup C_2$, then at the very least, the members of C_1 and C_2 can behave as if the merger did not take place and receive at least $v(C_1) + v(C_2)$ in total (Sandholm et al., 1999). However, larger coalitions often incur higher overheads in terms of coordination costs, hence not all coalitional games are superadditive. Note that if v is superadditive and non-negative (i.e., $v(C) \geq 0$, for all $C \subseteq N$), then it is monotonic; however, the reverse is not necessarily true.

Definition 7 (convex)**.** *A characteristic function game* ν *is* convex *if for all* $C, S \subseteq N$, $\nu(C \cup S) + \nu(C \cap S) \geq \nu(C) + \nu(S)$.

An equivalent characterization of convex coalitional games is: for all $i \in N$ and $S \subseteq T \subseteq N \setminus \{i\}$,

$$\nu(S \cup \{i\}) - \nu(S) \leq \nu(T \cup \{i\}) - \nu(T).$$

The value $\nu(C \cup \{i\}) - \nu(C)$ for $C \subseteq N \setminus \{i\}$ is the marginal contribution of i to coalition C. In convex coalitional games, an agent's marginal contribution increases with the size of the coalition (Jianhua, 1988; Rapoport, 1970; Shoham and Leyton-Brown, 2009).

Definition 8 (simple)**.** *A coalitional game is* simple *if the range of* ν *is* $\{0, 1\}$.

Some authors also require that simple coalitional games be monotonic (e.g., Bachrach et al. (2009a)). In simple coalitional games, the set of coalitions is partitioned into winning and losing sets. Winning coalitions are those that receive an outcome of 1, while losing coalitions are those that receive an outcome of 0. Simple coalitional games have been used to model voting situations (Elkind et al., 2008).

II.2 Solution Concepts

Game theoretic research on cooperation has focused mainly on how to distribute the collective earnings of the agents. As such, in this section agents are assumed to be self-interested. Various solution concepts have been proposed indicating which distributions, possibly unique or sets of distributions, rational agents will adopt. Roughly speaking these various solution concepts can be categorized as either stability related or fairness related. In this section, a brief overview of solution concepts in coalitional games is provided. Examples are provided illustrating important concepts and observations. The definitions of the solution concepts presented in this section come from the following game theory and multi-agent systems texts: Jianhua (1988), Rapoport (1970), and Shoham and Brown-Leyton (2009).

Important to all coalitional game solution concepts is the notion of a payoff allocation.

Definition 9 (payoff allocation)**.** *A* payoff allocation*,* payoff vector*, or simply an* allocation*, is a vector $q \in \mathbb{R}^n$ where $q_i \geq 0$ for all $i = 1, \cdots, n$ (where q_i is the i-th component of the vector q).*

q_i is interpreted as the utility allocated to agent i. For a coalition of agents $C \subseteq N$, let $q(C)$ denote $\sum_{i \in C} q_i$. That is, $q(C)$ is the collective earnings of all the agents in C.

Definition 10 (imputation)**.** *Given an n player coalitional game ν, an* imputation *is an allocation p such that:*

1. *p is efficient: $p(N) = \nu(N)$ and*
2. *p is individually rational: $\forall i \in N$, $p_i \geq \nu(i)$.*

Imputations are the payoff allocations that distribute the total earnings of the grand coalition and allocate to each agent at least the amount the agent earns on its own.

II.2.1 Stability Related Solution Concepts

Games Without Coalition Structures

One purpose of payoff allocation is to ensure stability. Intuitively, a proposed allocation is stable if and only if every coalition of agents receives at least as much in the allocation as it is capable of earning on its own (i.e., for all $C \subseteq N$, $p(C) \geq \nu(C)$). If some coalition of agents C can receive more utility than what it has been allocated by working with the rest of the agents (i.e., $\nu(C) > p(C)$), then each of the agents in C can profit by deviating from the current coalition structure by forming C and distributing the gain in utility equally amongst themselves (i.e., each agent can receive $p_i + \frac{\nu(C) - p(C)}{|C|}$). This observation leads directly to one of the most important stability related solution concepts (Jianhua, 1988; Rapoport, 1970; Shoham and Brown-Leyton, 2009).

Definition 11 (core)**.** *The* core *of a coalitional game ν is defined to be:*

$$core(\nu) = \{p \in \mathbb{R}^n : p(N) = \nu(N) \text{ and } \forall C \subseteq N \;\; p(C) \geq \nu(C)\}.$$

The core of a game ν is the set of all imputations that satisfy coalitional rationality. Intuitively, the core of a game ν is the set of all payoff vectors such that no coalition of agents can profitably break away from the grand coalition.

Definition 12 (excess)**.** *The* excess $e(p,C)$ *of a coalition C under a payoff allocation p is defined as* $p(C) - \nu(C)$.

An equivalent characterization of the core is the set of imputations p such that $e(p,C) \geq 0$ for all $C \subseteq N$.

The core of a game represents the set of completely stable payoff vectors (i.e., those allocations in which no coalition of agents can profitably deviate). Unfortunately, the core can be empty, meaning that there are no completely stable payoff vectors. Example 1 represents a game with an empty core.

Example 1. *Consider a setting with three agents and let ν be defined as follows:*

$$\nu(C) = \begin{cases} 1 & \text{if } |C| = 1 \\ 3 & \text{if } |C| = 2 \\ 4 & \text{if } |C| = 3. \end{cases}$$

Assume that $p \in core(\nu)$, then $p(\{1\}) \geq 1$ and $p(\{2,3\}) \geq 3$. Since $p(\{1,2,3\}) = 4$, it must be that $p(\{1\}) = 1$ and $p(\{2,3\}) = 3$. Thus, either $p(\{2\})$ or $p(\{3\})$ is at most $\frac{3}{2}$. Assume without loss of generality that $p(\{2\}) \leq \frac{3}{2}$. Thus, $p(\{1,2\}) = 1 + \frac{3}{2} = \frac{5}{2} < 3 = \nu(\{1,2\})$; contradicting the assumption that p is in $core(\nu)$. Therefore, $core(\nu)$ is empty.

Some sufficient conditions for the non-emptiness of the core are known (Jianhua, 1988; Rapoport, 1970; Shoham and Brown-Leyton, 2009). For example, if ν is convex, then $core(\nu)$ is not empty.

Bachrach et al. (Bachrach et al., 2009a,b) and Resnick et al. (Resnick et al., 2009) study the notion of cost of stability.

Definition 13 (cost of stability)**.** *The* cost of stability (CoS) *of a coalitional game* $G = (N, v)$ *is the minimum value of* Δ*, such that* $G(\Delta)$ *has a nonempty core, where* $G(\Delta) = (N, v')$ *and:*

$$v'(C) = \begin{cases} v(C) + \Delta & \text{if } C = N \\ v(C) & \text{otherwise.} \end{cases}$$

Intuitively, a coalitional game's cost of stability is the minimum external payment that can be made to the grand coalition for the game to be stable (i.e., have a non-empty core).

Even if the core is empty, it may be possible to distribute the utility of the grand coalition in such a way that no coalition can profit much by breaking away. This idea leads to the notion of an ε-core.

Definition 14 (ε-core)**.** *The* ε-core *of a coalitional game* v *is defined as:*

$$\varepsilon\text{-}core(v) = \{p \in \mathbb{R}^n : p(N) = v(N) \text{ and } \forall C \subseteq N \;\; p(C) \geq v(C) - \varepsilon\}.$$

Clearly, for sufficiently large values of ε, the ε-core will be nonempty (e.g., $\varepsilon = \max_{C \subseteq N} v(C)$). In general; however, smaller values of ε are preferred, as they indicate higher levels of stability.

Desiring smaller values of ε leads naturally to the least-core of a game. The least-core of a coalitional game is the set of payoff vectors that minimize the maximum deficit over all coalitions.

Definition 15 (least-core)**.** *The* least-core *of a coalitional game* v *is the intersection of all non-empty* ε*-cores of* v*:*

$$least\text{-}core(v) = \bigcap_{\varepsilon\text{-}core(v) \neq \emptyset} \varepsilon\text{-}core(v).$$

Let $e(v)$ be the least ε such that ε-$core(v)$ is non-empty.

Definition 16 ($e(v)$)**.**

$$e(v) = \inf\{\varepsilon \in \mathbb{R} : \varepsilon\text{-}core(v) \neq \emptyset\}.$$

A natural extension of the least-core solution concept is the nucleolus. While the least-core minimizes only the maximum deficit, the nucleolus continues by minimizing the second maximum deficit and so on.

Definition 17 (nucleolus)**.** *Let (N, v) be a coalitional game and let p be a payoff allocation. Let $\Theta(p)$ be the vector of excesses of coalitions under p, ordered non-increasingly. The nucleolus is the imputation that lexicographically minimizes $\Theta(p)$.*

The nucleolus always exists and is unique. Also, the nucleolus is always in the least-core.

Games With Coalition Structures

Up to this point, only situations in which the grand coalition forms have been considered. The definition of an imputation (and hence the core, ε-core and least-core) explicitly states that the utility of the grand coalition be distributed between the agents. However, in general, it may be desirable for the agents to form a number of disjoint coalitions, rather than to form the grand coalition. In such a case, an analog of the core for games where coalition structures are allowed can be defined (Elkind et al., 2008; Weiss, 1999).

Some authors, when considering coalitional games with coalition structures, fix a coalition structure and consider imputations with respect to this fixed coalition structure. This dissertation considers a slightly more general definition that considers all possible coalition structures simultaneously.

Definition 18 (CS-core)**.** *The* CS-core *of a coalitional game v is defined to be:*

$$CS\text{-}core(v) = \{(CS, p) \in \mathscr{P} \times \mathbb{R}^n : \forall C \in CS,\ \ p(C) = v(C) \text{ and } \forall C \subseteq N,\ \ p(C) \geq v(C)\}.$$

Instead of consisting of allocations like the core, the CS-core consists of pairs of coalition structures and allocations. The pair (CS, p) is in the CS-core if:

1. p distributes the utility of each coalition C in CS among the members of C: $p(C) = v(C)$.
2. No coalition of agents can profitably deviate: $\forall C \subseteq N$, $p(C) \geq v(C)$.

If p distributes the utility of each coalition in some coalition structure CS among that coalition's members, it is said that p is a payoff allocation (imputation) for CS.

If $core(v)$ is not empty, then $CS\text{-}core(v)$ is also not empty. In particular, if $p \in core(v)$, then $(\{N\}, p) \in CS\text{-}core(v)$ (Elkind et al., 2008). However, the CS-core can be non-empty when the core is empty, as Example 2 demonstrates.

Example 2. *Consider a setting with two agents and for each coalition C, let* v *be defined as follows:*

$$v(C) = 1 \quad \forall\, C \subseteq N.$$

$core(v)$ *is empty, since each agent on their own can obtain a value of* 1. $(\{\{1\},\{2\}\},(1,1))$ *is in* $CS\text{-}core(v)$*, since the only possible deviation is for* 1 *and* 2 *to form the coalition* $\{1,2\}$*, but this deviation will decrease the total earnings of the agents from* 2 *to* 1.

As with the core, Example 3 demonstrates that the CS-core of a game can also be empty.

Example 3. *Consider a setting with three agents and let* v *be defined as follows:*

$$v(C) = \begin{cases} 0 & \text{if } |C| = 1 \\ 1 & \text{if } |C| = 2 \\ 0 & \text{if } |C| = 3. \end{cases}$$

Assume that $CS\text{-}core(v)$ *is not empty. Clearly, the grand coalition and the coalition structure where all agents are in coalitions of size one are not in the CS-core, as any coalition of size two can deviate and increase their collective earnings from* 0 *to* 1. *Thus, by symmetry,*

the only remaining coalition structure to consider is $\{\{1,2\},\{3\}\}$. *Since agents* 1 *and* 2 *must divide a utility of* 1 *between themselves, at least one agent must receive no more than* $\frac{1}{2}$. *However, this agent can form a coalition with agent* 3 *and receive a collective utility of* $1 > \frac{1}{2}$. *Thus,* $CS\text{-}core(v) = \emptyset$.

While corresponding notions of the ε-CS-core, least-CS-core, and CS-nucleolus do not seem to have been defined previously, all three provide natural measures of stability for games with an empty CS-core. The definitions of the ε-CS-core, the least-CS-core, and the CS-nucleolus are analogous to the definitions of the ε-core, least core, and nucleolus, respectively.

Definition 19 (ε-CS-core)**.** *The* ε-CS-core *of a coalitional game* v *is defined as:*

$$\varepsilon\text{-}core(v) = \{(CS,p) \in \mathscr{P} \times \mathbb{R}^n : \forall C \in CS,\ \ p(C) = v(C) \text{ and } \forall C \subseteq N,\ \ p(C) \geq v(C) - \varepsilon\}.$$

Definition 20 (least-CS-core)**.** *The* least-CS-core *of a coalitional game* v *is the intersection of all non-empty* ε*-CS-cores of* v*:*

$$least\text{-}CS\text{-}core(v) = \bigcap_{\varepsilon\text{-}CS\text{-}core(v) \neq \emptyset} \varepsilon\text{-}CS\text{-}core(v).$$

As with games without coalition structures, let $e(CS,v)$ be the least ε, such that there exists an imputation p for the coalition structure CS, such that $(CS,p) \in \varepsilon\text{-}core(v)$.

Definition 21 ($e(CS,v)$)**.**

$$e(CS,v) = \inf\{\varepsilon \in \mathbb{R} : \exists p\ \ (CS,p) \in \varepsilon\text{-}core(v)\}.$$

Definition 22 (CS-nucleolus)**.** *Let* (N,v) *be a coalitional game, CS a coalition structure, and let* p *be a payoff allocation for CS. Let* $\Theta(p)$ *be the vector of excesses of coalitions under* p *ordered non-increasingly. The CS-nucleolus consists of those pairs* (CS,p) *such*

that p lexicographically minimizes $\Theta(p)$.

While the nucleolus of a coalitional game is always unique, the CS-nucleolus is highly non-unique. If (N, v) is any additive coalitional game, then for every coalition structure CS, the vector p defined by

$$p_i = v(\{i\})$$

is an imputation for CS and $(CS, p) \in CS\text{-}nucleolus(v)$.

II.2.2 A Fairness Related Solution Concept

Another purpose of payoff allocation is to distribute the collective earnings in a fair way. Fairness related solution concepts, as their name suggests, attempt to allocate the collective utility in such a way as to satisfy certain intuitive notions of fairness. However, the allocations produced by such solution concepts may not be the most stable. One such solution concept is presented in this subsection.

The Shapley value allocates to each agent i the agent's average marginal contribution to the grand coalition, averaged over all possible permutations of the agents (i.e., averaged over all possible orderings in which the agents can join the grand coalition) (Jianhua, 1988; Rapoport, 1970; Shoham and Brown-Leyton, 2009).

Definition 23 (Shapley value)**.** *The* Shapley value *of an agent i,* $\phi_i(v)$*, is defined as:*

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left(v(S \cup \{i\}) - v(S)\right).$$

The Shapley value seems to be intuitively fair, since each agent is allocated the average of its marginal contribution to the grand coalition. The Shapley value can be shown to be the unique payoff allocation that satisfies certain formalized notions of fairness. The Shapley value is the only payoff allocation method that simultaneously satisfies the following axioms:

1. Individual rationality $\phi_i(v) \geq v(\{i\})$: Each agent obtains at least as much as the agent can obtain on its own.

2. Efficiency $\sum_{i \in N} \phi_i(v) = v(N)$: The total earnings of the agents is distributed.

3. Symmetry: If for all $S \subseteq N - \{i, j\}$, $v(S \cup \{i\}) = v(S \cup \{j\})$, then $\phi_i(v) = \phi_j(v)$.

4. Additivity: If v and ω are both characteristic functions on N, then $\phi_i(v + \omega) = \phi_i(v) + \phi_i(\omega)$.

II.3 Compact Representations of Coalitional Games

The naive representation of a coalitional game requires specifying $\Theta(2^n)$ numbers. However, when considering the computational complexity of computing various game theoretic solution concepts, it is more natural and desirable to measure the complexity of the problem in terms of compact representation (Deng and Papadimitriou, 1994). That is, many computational questions are trivially polynomial time solvable when the input consists of $\Theta(2^n)$ values.

One commonly studied compact coalition game representation are weighted voting games (Bachrach et al., 2009a; Elkind et al., 2008; Shoham and Brown-Leyton, 2009; Zuckerman et al., 2008). In a weighted voting game, each agent i is assigned a real-valued weight w_i. A coalition of agents C achieves a value of 1 or "wins", if the sum of the weights of its agents meets or exceeds a given quota q. Otherwise the coalition C achieves a value of 0 or "loses".

Definition 24. *A* weighted voting game *is a tuple* $[q; w_1, \cdots, w_n]$ *where* q *is the quota and* w_i *is the weight of agent* i *for* $i = 1, \cdots, n = |N|$. *The value of a coalition* $C \subseteq N$ *is defined as:*

$$v(C) = \begin{cases} 1 & \text{if } \sum_{i \in C} w_i \geq q \\ 0 & \text{otherwise.} \end{cases}$$

k-weighted voting games are an extension of weighted voting games.

Definition 25. *A* k-weighted voting game*, with characteristic function* ν*, is a* k*-tuple* $(w_1, \cdots, w_k)$ *where each* w_i*,* $i = 1, \cdots, k$ *is a weighted voting game. A coalition of agents* C *receives a value of* 1 *if and only if it receives a value of* 1 *in each of the* k *weighted voting games. That is, if* ν_i *is the characteristic function for the weighted voting game* w_i *then:*

$$\nu(C) = \begin{cases} 1 & \text{if } \forall i = 1, 2, \cdots, k, \ \ \nu_i(C) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

k-weighted voting games are a *complete* representation for simple coalitional games. That is, each simple coalitional game can be represented as a k-weighted voting game.

Ohta et al. (Ohta et al., 2009) consider the complexity of the coalition structure generation problem under three compact coalitional game representations. They show that in each case, the problem of finding an optimal coalition structure is NP-hard.

One of the more natural compact representations considered by Ohta et al. (Ohta et al., 2009) is the marginal contribution network that assigns values to coalitions based upon which groups of agents are present and not present in the coalition.

Definition 26 (marginal contribution network)**.** *A* marginal contribution network (MC-net) *is a set of rules* R*. Each rule* r *consists of two disjoint subsets of agents* P_r *and* N_r *and a value* v_r*. The rule* r *is applicable to a coalition* C *of agents if every agent in* P_r *is also in* C*,* $P_r \subseteq C$*, and every agent in* N_r *is not in* C*,* $N_r \cap C = \emptyset$*. For a coalition* C*, let* R_C *be the set of rules applicable to* C*. The value* $\nu(C)$ *is given by* $\sum_{r \in R_C} v_r$*.*

One important feature of MC-nets is that they are a complete representation of coalitional games. That is, every coalitional game can be represented as a MC-net. Given an arbitrary characteristic function ν, define a rule r_C for every coalition of agents C as follows: $P_{r_C} = C$, $N_{r_C} = N \setminus C$ and $v_{r_C} = \nu(C)$.

A number of compactly representable coalitional games are defined on graphs. Given a graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}$, a coalitional game ν can be defined

where the set of agents coincides with the vertex set V, and the value of a coalition $C \subseteq V$ is the sum of the weights of the edges in the subgraph induced by C.

Definition 27 (graph game)**.** *Given an undirected graph $G = (V,E)$ and a weight function $w : E \to \mathbb{R}$, a* graph game (N, ν) *is defined as $N = V$ and for $C \subset N$,*

$$\nu(C) = \sum_{u,v \in C} w(\{u,v\}).$$

That is, $\nu(C)$ is the sum of the edge weights in the subgraph induced by C.

Notice that graph games are one of the few situations in this dissertation when negative coalitional values are permitted. The following variant of graph games is commonly studied.

Definition 28 (positive graph game)**.** *Given an undirected graph $G = (V,E)$ and a weight function $w : E \to \mathbb{R}^+$, a* positive graph game (N, ν) *is defined as $N = V$ and for $C \subset N$,*

$$\nu(C) = \sum_{u,v \in C} w(\{u,v\}).$$

That is, $\nu(C)$ is the sum of the edge weights in the subgraph induced by C.

Thus, a positive graph game is a graph game where all the edge weights are nonnegative. Positive graph games are always super additive.

Another way to define a coalitional game on an unweighted graph, $G = (V,E)$, is to set the value of a coalition $C \subseteq V$ to be the size of a maximum independent set in the graph induced by C.

Definition 29 (independent set game)**.** *Given an undirected graph $G = (V,E)$, an* independent set game (N, ν) *is defined as $N = V$ and for $C \subset N$, $\nu(C)$ is equal to the number of vertices in a maximum independent set in the subgraph induced by C.*

Recall that a flow network is defined by an edge weighted directed graph along with two distinguished vertices, the source, s, and sink, t.

Definition 30 (flow network)**.** *A* flow network $F = (V, E, c, s, t)$ *is a* 5*-tuple where* (V, E) *is a directed graph,* $c : E \to \mathbb{R}^+$ *is a function mapping edges to positive capacities and* $s, t \in V$ *are two distinguished vertices.*

Given a flow network F, a coalitional game (N, v), where $N = E$, can be created by defining the value of a coalition $C \subset E$ to be the maximum flow through the network induced by C.

Definition 31 (network flow game)**.** *Given a flow network* $F = (V, E, c, s, t)$*, a* network flow game (N, v) *is defined as* $N = E$ *and for each* $C \subset N$*,* $v(C)$ *is equal to the maximum flow through the flow network* (V, C, c', s, t)*, where* c' *is the function* c *restricted to edges in* C*.*

Given an weighted undirected graph $G = (V, E)$, a coalitional game (N, v), where $N = V$, can be created by defining $v(C)$ to be the weight of a maximum matching on the subgraph induced by $C \subseteq V$.

Definition 32 (matching game)**.** *Given an undirected graph* $G = (V, E)$ *and a weight function* $w : E \to \mathbb{R}$*, a* matching game (N, v) *is defined as* $N = V$ *and for each* $C \subset N$*,* $v(C)$ *is the weight of a maximum matching on the subgraph induced by* C*.*

This dissertation considers one last compact representation.

Definition 33 (coalitional skill game)**.** *Let* N *be a set of agents,* S *a set of* skills*, and* T *a set of* tasks*. Let* $u : 2^T \to \mathbb{R}^+$ *be a monotonic function. Each agent* $i \in N$ *has an associated set of skills* $S(i) \subseteq S$ *and each task* $t \in T$ *has an associated required skill set* $S(t) \subseteq S$*. For a coalition* $C \subseteq N$*, let* $S(C) = \cup_{i \in C} S(i)$ *be the set of skills possessed by the* C*. The set of tasks a coalition* C *can perform is* $T(C) = \{t \in T : S(t) \subseteq S(C)\}$*. A* coalitional skill game (N, v) *is defined as* $v(C) = u(T(C))$ *for all* $C \subseteq N$*.*

II.4 Complexity Theory

Knowledge of standard definitions and complexity classes from computational complexity theory is assumed. In particular, it is assumed that the reader is familiarly with the

complexity classes P and NP and the notions of NP-hardness and NP-completeness (Karp, 1972; Garey and Johnson, 1990).

In many computational problems the complexity of the problem is dependent upon one particular aspect of the input. If that particular aspect of the input is small, the problem can often be solved efficiently. The notion of parameterized complexity formalizes such situations.

Definition 34 (parameterized problem)**.** *Let Σ be a finite alphabet. A* parameterized problem *is a subset $L \subseteq \Sigma^* \times \mathbb{N}$.*

Definition 35 (fixed-parameter tractable)**.** *A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is* fixed-parameter tractable *if membership of (x,k) in L can be determined in time $f(k) \cdot |x|^{O(1)}$, where $|x|$ is the length of the string x.*

As a motivating example of parameterized complexity, consider the vertex cover problem. Recall that a vertex cover in a graph $G = (V,E)$ is a set of vertices $S \subset V$ such that every edge in E is incident to at least one vertex in S. Finding a smallest vertex cover is NP-complete. If G has a vertex cover consisting of k vertices, then the vertex cover problem can be solved in $2^k n^{O(1)}$ time. Thus, vertex cover is fixed parameter tractable in the size of the minimal vertex cover.

The vast majority of the algorithms presented in this dissertation have a runtime exponential in the number of agents, n. Rather than present and prove cumbersome runtime bounds involving factors polynomial in n, the presentation of algorithmic results are simplified by employing O^* notation. Formally, an algorithm is said to run in time $O^*(f(n))$, for some computable function $f(n)$ if it runs in time $O(f(n)\log^c(f(n))$ for some constant c. Thus, O^* notation explicitly ignores factors that are polylogarithmic in the bound $f(n)$. For example, the use of O^* notation allows the simplification of runtime bounds such as $O(n^2\log(n)2^n)$ time to $O^*(2^n)$. When comparing exponential time algorithms, it is the base of the exponent that influences the runtime the greatest. Hence, differences in runtime

bounds can be more easily compared by dropping factors polynomial in n. An analogous definition applies to o^*.

CHAPTER III

Literature Review

If you want to be incrementally better: Be competitive. If you want to be exponentially better: Be cooperative.[1]

III.1 Coalition Formation

Agents or individuals may cooperate for many reasons. In general, the reasons agents choose to cooperate depends on whether they are self-interested or group rational. Self-interested agents will choose to cooperate only if it is of benefit to them. For example, self-interested agents may cooperate in order to achieve goals that they are unable to achieve on their own. On the other hand, group rational agents attempt to do that which is best for the group. Such agents cooperate whenever it increases the collective utility of the group, possibly even at the expense of the individual agent's own utility.

Cooperation between agents, either self-interested or group rational, involves four activities (Rahwan, 2007; Sandholm et al., 1999; Weiss, 1999):

1. Coalition value calculation: Determining, or estimating, the value of each coalition.
2. Coalition structure generation: Determining which coalitions will form.
3. Earning the utility: Coalitions of agents work as a team to guarantee some utility for themselves.
4. Dividing the earnings: Once a coalition has earned its utilities, it is necessary to divide the earnings amongst the member agents.

These activities can be interrelated, especially coalition structure generation and payoff allocation. A self-interested agent will agree to a coalition structure only if awarded a

[1] Unknown source - `http://en.proverbia.net/citastema.asp?tematica=258`.

fair share of the earnings. The earnings an agent is awarded is directly related to that agent's contribution to the earnings of the coalition in which it is a member. Likewise, how agents earn their joint utility is related to the activity of coalition value calculation, as it is necessary to know how coalitions earn their utility in order to estimate how much utility they will earn.

Each of the four activities of multi-agent cooperation and coalition formation are discussed in the following subsections. The contributions of this dissertation to each activity are also provided.

III.1.1 Coalition Value Calculation

Before agents can determine which coalitions to form, they must first know the relative value of each potential coalition. Typically, the coalition value calculation involves determining the amount of utility each coalition of agents can earn. A recent algorithm developed by Rawhan and Jennings (Rahwan, 2007; Rahwan and Jennings, 2005) permits efficient distributed and decentralized coalition value calculation with no communication required between agents. Since algorithms already exist that distribute the coalition value calculations among agents efficiently, this dissertation does not focus on this aspect of multi-agent cooperation.

III.1.2 Coalition Structure Generation

For group rational agents, the most desirable coalition structure is one that maximizes the total social welfare of the group. The social welfare of a coalition can be defined in a number of different ways. A utilitarian definition of social welfare may be the sum of the utilities of all coalitions. An egalitarian definition may be the minimum utility of all coalitions.

Much research on coalition structure generation in multi-agent systems assumes group rational agents (Michalak et al., 2009a; Rahwan and Jennings, 2007; Rahwan et al., 2009b; Sandholm et al., 1999; Service and Adams, 2010a,b, 2011b; Larson and Sandholm, 2000).

Unfortunately, with black-box access to the value function, the current fastest algorithm for determining the optimal coalition structure for a group of n agents runs in $O^*(3^n)$ time (Sandholm et al., 1999; Rahwan and Jennings, 2008b; Yeh, 1986). Due to the high computational complexity of coalition structure generation, recent research has turned to the development of anytime algorithms and heuristic techniques (Sandholm et al., 1999; Rahwan et al., 2009b; Service and Adams, 2011b; Sen and Dutta, 2000). Unfortunately, the algorithms developed in these two classes have drawbacks. Heuristic algorithms are generally not capable of providing guarantees on the quality of the solutions they produce. Current anytime algorithms are capable of producing quality guarantees, but have a vastly worse worst case runtime then the fastest design-to-time algorithms (the best bound known on the running time of all current anytime algorithms to find the optimal solution is $O^*(n^n)$). Even generating a solution that is guaranteed to be close to the optimal (e.g., within any constant factor of the optimal) with current anytime algorithms requires, in the worst case, $\Omega(3^n)$ time.

This dissertation addresses the lack of coalition structure generation algorithms with good theoretical guarantees in the following ways:

1. The first algorithms capable of guaranteeing constant factor approximations to the optimal coalition structure are presented. For example, the presented approximation algorithm for general coalitional games is capable of guaranteeing a solution that is within a factor of $\frac{2}{3}$ of the optimal in $O^*(2.83^n)$ time, a solution within $\frac{1}{2}$ of the optimal in $O^*(2.59^n)$ and a solution that is within a factor of $\frac{1}{4}$ of the optimal in $O^*(2^n)$ time (intermediate approximation guarantees, between $\frac{1}{4}$ and $\frac{1}{2}$ can be generated as well). The $\frac{1}{4}$ approximation is optimal in the sense that no algorithm can guarantee a $\frac{1}{4}$ approximation solution in less asymptotic time, up to polynomial factors.

2. This dissertation also presents the first use of randomization in the coalition structure generation process. Randomized approximation algorithms are presented that are capable of generating the same quality bounds as their deterministic counterparts in

less asymptotic time.

3. For general coalitional games, it is known that $\Omega(2^n)$ time is required to provide any guarantees on the quality of the solution produced. This dissertation is the first to study coalition structure generation in the class of monotonic coalitional games. It is demonstrated that in monotonic games the general lower bound result no longer holds. An algorithm is presented for obtaining various performance guarantees in $o(2^n)$ time. For example, the algorithm presented in Chapter IV.3 can obtain a $\sqrt{n}$ factor approximation in $O^*((1+\varepsilon)^n 2^{\sqrt{n}\log n})$ time, for all $\varepsilon > 0$.

III.1.3 Earning the Joint Utility

Many commonly studied domains model the coalition formation problem with the notion of a task the agents are assigned to complete (Abdallah and Lesser, 2004; Service, 2009; Service and Adams, 2011a; Shehory and Kraus, 1995, 1996, 1998; Vig and Adams, 2005, 2006; Vig, 2006; Vig and Adams, 2007). Other work almost completely ignores how the agents collectively earn their utility, assuming that for each coalition it is known in advance how much utility can be received (Rahwan and Jennings, 2008a,b; Rahwan et al., 2009b; Sandholm et al., 1999; Service and Adams, 2010a,b, 2011b). This dissertation follows the latter approach. Such an abstraction permits focusing on *which* coalitions of agents forms and *why* individual self-interested agents agree to said coalitions without becoming lost in the details of *how* coalitions of agents earn their utility.

III.1.4 Payoff Allocation

Payoff allocation is generally only of interest when agents are self-interested (a group-rational agent's only goal is to maximize the social welfare of the group). The payoff allocated to each agent serves two purposes: The first is to fairly award an agent for its contributions to the coalition. The second purpose is to disallow profitable deviations from the agreed upon coalition structure. Intuitively, a payoff allocation is stable if no coalition

of agents can earn more working together than they collectively earn under the current payoff allocation. In such situations, no coalition of agents has an incentive to deviate as it will result in a decrease in earnings for the agents involved.

Typical research in game theory and economics on coalitional games focuses on the payoff allocation aspect (Jianhua, 1988; Rapoport, 1970). Different solution concepts have been proposed, emphasizing either the fairness or the stability of the resulting payoff allocation.

However, unlike competitive game theory research on the price of anarchy, little research has investigated the relationship between optimality and stability. When agents are self-interested, rather than group rational, social welfare maximizing coalition structures are not necessarily the most preferable coalition structures. Brânzei and Larson (Brânzei and Larson, 2009) show that in non-transferable games, a class of coalitional games that restrict how the agents may divide the collective earnings amongst themselves, social welfare maximizing coalitions may not be stable, while some non-social welfare maximizing coalitions can be completely stable.

In the case considered in this dissertation of transferable coalitional games (games in which there are no restrictions on how the agents may divide the utility amongst themselves), even less research relating optimality and stability has been published. Indeed the only mention of both optimality and stability in the same context seems to be in a passing remark that only social welfare maximizing coalition structures can be in the CS-core (Weiss, 1999). While no proof of this fact is provided in Weiss (1999), it is a straight forward corollary of Theorem 10 provided by Aumann and Dreze (1974) that only social welfare maximizing coalition structures can be in the CS-core. However, Aumann and Dreze do not explicitly mention this corollary or further explore the coalition structure stability-optimality relationship.

This dissertation digs deeper into the relationship between optimality and stability and demonstrates that the relationship between the two deserves more than a passing remark.

It is proven in Chapter VII.1 that if the CS-core of a game is not empty, then every social welfare maximizing coalition structure is in the CS-core and that all social welfare maximizing coalition structures are payoff equivalent (i.e., if (CS_1, p) is the in the CS-core, then for any other optimal coalition structure CS_2, (CS_2, p) is in the CS-core as well). From the perspective of a self-interested agent, all social-welfare maximizing coalition structures are equally desirable, as the set of payoffs the agent can receive is the same in each. This result provides a complete characterization for the relationship between optimality and stability in coalitional games with a non-empty *CS-core*.

Many games of interest; however, have an empty CS-core. In such situations, it is shown that the optimality-stability relationship becomes more complex. In particular, this dissertation examines the optimality-stability relationship with respect to the least-CS-core solution concept. The following observations are made:

1. There are coalitional games such that only non-social welfare maximizing coalition structures are in the least-CS-core.

2. There are coalitional games such that some coalition structures in the least-CS-core are social welfare maximizing, while others are not.

3. If a coalitional game is superadditive, then the least-CS-core consists of only social welfare maximizing coalition structures. However, not necessarily all optimal coalition structures will be present in the least-core, nor will those that are in the least-CS-core be payoff equivalent.

This dissertation also shows that in a number of compactly representable classes of coalitional games, only optimal coalition structures can be maximally stable under the *least-CS-core* solution concept. It is also shown; however, that in some classes some sub-optimal coalition structures may be more stable than all optimal coalition structures.

III.2 Algorithms for Coalition Structure Generation

This section presents a survey of the research results and algorithms for the coalition structure generation problem. Algorithms from three different classes are surveyed. Throughout this section, agents are assumed to be group rational.

III.2.1 Impossibility Result

Before surveying various algorithms for coalition structure generation, a lower bound is provided on the runtime of any coalition structure generation algorithm that provides guarantees on the quality of the solutions it produces. It is relatively easy to see that in order to generate a coalition structure guaranteed to be within any factor of the optimal it is necessary to observe the value of each coalition of agents at least once (Sandholm et al., 1999). This result follows, as the value of an unseen coalition can be arbitrarily large. As there are $2^n - 1$ possible coalitions given n agents, there is an immediate lower bound of $\Omega(2^n)$ for the running time of any coalition structure generation algorithm that provides guarantees on the quality of the solutions it produces.

III.2.2 Design-to-time Algorithms

Design-to-time algorithms are guaranteed to find an optimal solution; however, these algorithms must be run to completion in order to do so. Design-to-time algorithms are unable to produce intermediate solutions, like anytime algorithms. However, this deficit is compensated for in that design-to-time algorithms have a better worst case runtime than the current anytime algorithms. Currently the fastest design-to-time algorithms for coalition structure generation employ dynamic programming to determine an optimal solution in $O^*(3^n)$ time for n agents (Rahwan and Jennings, 2008b; Sandholm et al., 1999; Yeh, 1986).

The following notation will be used for the discussion of the standard dynamic programming algorithm for coalition structure generation:

1. Let $opt(C)$ denote the *maximum value* over all coalition structures that can be formed

from the agents in C.

2. Let $sol(C)$ denote the *optimal coalition structure* for the subproblem consisting of only those agents in C.

Algorithm 1 provides pseudo code for the coalition structure generation dynamic programming algorithm (Rahwan and Jennings, 2008b; Sandholm et al., 1999; Yeh, 1986).

Algorithm 1 Dynamic Programming Algorithm

```
1: for k = 1 to n do
2:   for C ⊆ N, |C| = k do
3:     opt(C) ← v(C)
4:     sol(C) ← {C}
5:     for C' ⊂ C do
6:       if opt(C') + opt(C \ C') > opt(C) then
7:         opt(C) ← opt(C') + opt(C \ C')
8:         sol(C) ← sol(C') ∪ sol(C \ C')
9:       end if
10:    end for
11:  end for
12: end for
13: return sol(N)
```

Algorithm 1 is based on the observation that if C is a coalition in an optimal solution CS to a subproblem $N' \subseteq N$, then $CS \setminus \{C\}$ is an optimal solution to the subproblem $N' \setminus C$. Observe that if there is a higher valued solution CS' to $N' \setminus C$, then $CS' \cup \{C\}$ is a better solution to the subproblem N' than CS, contradicting the fact that CS was an optimal solution to N'.

Algorithm 1 works by iteratively constructing the optimal solution to each subproblem (subset of agents) in the order of increasing size. Prior to constructing the solution to a subproblem $N' \subseteq N$, the optimal solution for each subproblem of N' has to be determined. When determining the optimal solution to the subproblem N', the dynamic programming algorithm considers each coalition in N' for possible inclusion in the solution to N'. Given a coalition $C \subseteq N'$, the highest valued solution to the subproblem N' to which C is a member is simply $\{C\} \cup sol(N' \setminus C)$. Taking a maximum over all subsets of N' requires $O(2^{|N'|})$

time and results in the optimal solution to N'.

The running time of Algorithm 1 can be easily bounded above by noting that there are $\binom{n}{k}$ coalitions of k agents and each requires $O^*(2^k)$ time. Summing over all possible coalition sizes yields the following time bound:

$$\sum_{k=1}^{n} \binom{n}{k} O^*(2^k) = O^*(3^n).$$

III.2.3 Anytime Algorithms

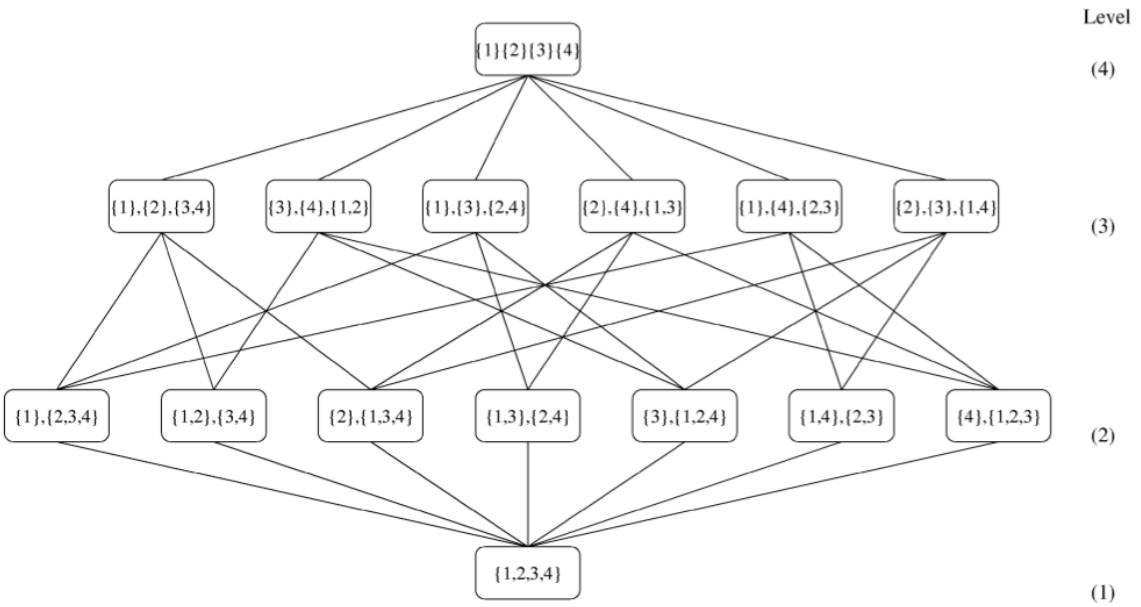
Anytime algorithms permit premature termination (i.e., before an optimal solution has been found), while also possibly providing guaranteed bounds on the quality of the solution found at each point during the search. For computationally expensive problems, like coalition structure generation, the ability to stop the search once a solution of sufficient quality has been found is desirable. However, one of the main shortcomings of current anytime algorithms for coalition structure generation is that all, in the worst case, require $O(n^n)$ time to find the optimal solution. Several anytime coalition structure generation algorithms are surveryed in this section.

Coalition Structure Graph Search

The first anytime coalition structure generation algorithm was developed by Sandholm et al. (Sandholm et al., 1999), who introduce the coalition structure graph (Figure III.1). In the coalition structure graph, each node represents a different coalition structure. Edges between two nodes (coalition structures) represent mergers of coalitions when followed down the graph and splittings of coalitions when followed up. For example, in the graph in Figure III.1, there is an edge between the nodes labeled $\{1\},\{3\},\{2,4\}$ and $\{1,3\},\{2,4\}$ because coalitions $\{1\}$ and $\{3\}$ can merge to form coalition $\{1,3\}$. Likewise coalition $\{1,3\}$ can split into the two coalitions $\{1\},\{3\}$.

Sandholm et al. (Sandholm et al., 1999) provide an anytime coalition structure genera-

Figure III.1: Coalition structure graph for a game with four agents. Each node represents a coalition structure. Edges between nodes represent mergers between coalitions when followed down and splittings of coalitions when followed up. Reproduced from Sandholm et al. (Sandholm et al., 1999).



tion algorithm that searches through nodes in the coalition structure graph. Their algorithm maintains the best coalition structure observed thus far and if terminated before the search has completed, simply returns this best-so-far solution. Their algorithm begins by searching the lowest two levels of the graph (i.e., those coalition structures that contain at most two coalitions). They prove that after the lowest two levels of the graph have been searched, then the current best observed solution is within a factor of n of the optimal solution.

After searching the lowest two levels of the graph, Sandholm et al.'s algorithm proceeds to perform a breadth first search of the coalition structure graph starting from the top most node. After searching the top most level of the graph, Sandholm et al. show that the bound on the quality of their current best solution drops from n to $\frac{n}{2}$. Sandholm et al. also show that after searching level l, the bound on the quality of the current solution will be either $\lceil \frac{n}{h} \rceil$ or $\lfloor \frac{n}{h} \rfloor$, depending on the number of agents, n, in the system and the level, l, searched,

where $h = \lfloor \frac{n-l}{2} \rfloor + 2$.

Unfortunately, in the worst case, in order for Sandholm et al.'s algorithm (Sandholm et al., 1999) to find the optimal solution, every coalition structure must be examined. Since the number of coalition structures is $\omega(n^{n/2})$ and $O(n^n)$ (Sandholm et al., 1999), the dynamic programming algorithm is significantly faster at finding the optimal solution.

Dang and Jennings' Algorithm

Dang and Jennings (Dang and Jennings, 2004) also present an anytime algorithm for coalition structure generation. Dang and Jennings' algorithm explores the coalition structures in the coalition structure graph in a different order from Sandholm et al.'s algorithm (Sandholm et al., 1999).

Like Sandholm et al.'s algorithm (Sandholm et al., 1999), Dang and Jennings' algorithm first explores the bottom two levels of the coalition structure graph, followed by the topmost level. After searching these three levels of the coalition structure graph, Dang and Jennings' algorithm explores specific subsets of the remaining levels based upon the sizes of the coalitions within them. Specifically, Dang and Jennings' algorithm searches those coalition structures that have between 3 and $n-1$ coalitions and have a coalition of size $\lceil n(q-1)/q \rceil$, where q is initially $\lfloor \frac{n+1}{4} \rfloor$ and is incrementally decreased to 2. They show that, for a particular q, after searching all such coalition structures, the best coalition structure observed so far is within a bound of $2q-1$ from the optimal.

Like Sandholm et al.'s algorithm (Sandholm et al., 1999), for Dang and Jenning's algorithm to find an optimal solution, in the worst case, all coalition structures must be examined.

Integer Partition Algorithm

Rahwan et al. (Rahwan et al., 2009b, 2007a,b; Michalak et al., 2010) demonstrated how branch and bound can be employed in an anytime coalition structure generation algorithm. Rahwan et al.'s algorithm partitions the space of coalition structures based upon the sizes

of their constituent coalitions. Each subspace corresponds to an integer partition of n, where n is the number of agents. For example, in a situation with four agents there are five subspaces: $[1,1,1,1]$, $[1,1,2]$, $[2,2]$, $[1,3]$ and $[4]$. The coalition structure $\{\{1,2\},\{3\},\{4\}\}$ belongs to the subspace $[1,1,2]$, while the coalition structure $\{\{1,2,3\},\{4\}\}$ belongs to the coalition structure $[1,3]$. Rahwan et al. (Rahwan et al., 2009b, 2007a,b) show how to bound the value of the solutions in each subspace.

Rahwan et al.'s algorithm (Rahwan et al., 2009b, 2007a,b) consists of two phases. First, a scan through all possible coalitions is performed. During this first phase, bounds are computed for each subspace and an initial solution is generated that is guaranteed to be within a factor of $\frac{n}{2}$ of the optimal. The value of this initial solution provides a lower bound on the value of an optimal coalition structure. Those subspaces with an upper bound less than the value of this initial solution are pruned from further consideration.

After the first phase is complete, Rahwan et al.'s algorithm begins searching individual subspaces by enumerating the coalition structures within them. As the algorithm searches, it remembers the best coalition structure observed so far. If a new coalition structure is found that is better than any previously seen coalition structure, then those subspaces that have an upper bound less than the value of the new best coalition structure are pruned. The search can end in one of two ways:

1. The search can end when all subspaces have either been pruned from consideration or have been searched, at which point an optimal coalition structure has been found.

2. The search can be terminated due to a lack of time, in which case the best coalition structure found thus far is returned. Even though all coalition structures have not been observed or pruned, a bound can still be placed on the relative value of this returned coalition structure compared to an optimal one. If UB is the largest upper bound of any unsearched subspace and V is the value of the returned coalition structure, then the ratio of V to the value of an optimal coalition structure V^*, $\frac{V}{V^*}$, is at least $\frac{V}{UB}$.

The method used to bound the value of the best solution discovered thus far immediately suggests an order in which to search the different subspaces. Subspaces are searched in order of decreasing upper bounds. By eliminating subspaces with larger upper bounds early in the search, better bounds on the quality of the current solution can be obtained. Rahwan et al. (Rahwan et al., 2009b) also suggest other possible search orders for situations where the user only desires a solution within say 90% of the optimal.

Rahwan et al.'s algorithm, unlike Sandholm et al. and Dang and Jennings' algorithms, does not have to examine every coalition structure. If the algorithm is capable of generating tight bounds on the quality of the solutions in any subspace, then many subspaces can be pruned from consideration and the coalition structures within the subspaces never examined.

Another fundamental difference between Rahwan et al.'s algorithm, Sandholm et al., and Dang and Jennings' algorithms is that Rahwan et al.'s algorithm generates bounds based solely upon the values of the coalitions. The bounds generated by previous coalition structure generation algorithms were based only on the amount of search performed and not the actual values of the coalitions and coalition structures observed.

Recently, Rahwan et al. (Rahwan et al., 2011) have investigated the minimum set of coalition structures whose value must be examined in order to be guaranteed to find a coalition structure within a given bound β of the optimal.

Improved Dynamic Programming - Integer Partition Algorithm

Rahwan et al. (Rahwan and Jennings, 2008a) also show how to combine their dynamic programming algorithm (Rahwan and Jennings, 2008b) (IDP) with their integer partition algorithm (Rahwan et al., 2009b, 2007a,b) (IP) in order to form a hybrid (IDP-IP) algorithm that explores part of the space through dynamic programming and part of the space through a branch and bound search in a reduced space of coalition structures. IDP-IP works by solving all subproblems that include up to m agents (where m is a parameter of

the algorithm) with IDP. After IDP has completed, IP is used to search through the space of coalition structures directly. However, many coalition structures no longer have to be searched when using solutions to the subproblems solved with IDP.

Let v be a characteristic function and for each coalition $C \subseteq N$, let $opt(C)$ be the value of an optimal partitioning of the agents in C. Define v' as follows:

$$v'(C) = \begin{cases} opt(C) & \text{if } |C| \leq m \\ v(C) & \text{if } |C| > m. \end{cases}$$

Intuitively, using IDP to determine the optimal solution to all subproblems of size m or less creates v' given v (i.e., after IDP computes the value of the optimal solution to all subproblems of size m or less are known). After IDP completes, IP searches the space of coalition structures using v', rather that v. Recall that IP partitions the space of coalition structures based upon the sizes of the coalitions within them and then searches the space using branch and bound. However, subspaces that contain coalition structures with two coalitions of total size no greater than m need not be searched. To observe this fact, let S_1 be any subspace that contains coalition structures with two coalitions of combined size no greater than m. Let k and j denote the sizes of those coalitions. Thus $k + j \leq m$. Let S_2 be the subspace that contains the same coalition sizes as S_1, except that k and j are replaced by $k + j$. That is, the coalition structures in S_2 contain one less coalition than do the coalition structures in S_1. Also, the sizes of the coalitions in every coalition structure in S_2 are the same as the sizes of the coalitions in the coalition structures in S_1 except that the coalition structures in S_2 contain a coalition of size $k + j$, where the coalition structures in S_1 contain two coalitions, one of size k and one of size j. Since $k + j \leq m$, IDP has already computed the optimal solution over all subproblems of size $k + j$. Thus, while searching S_2, IDP-IP is simultaneously searching S_1 as it considers coalitions of size $k + j$, and all possible partitionings of those $k + j$ agents.

Rahwan et al. (Rahwan and Jennings, 2008a) show IDP-IP to be empirically faster, for

certain values of the parameter m, than both IDP and IP. However, the worst case theoretical runtime of IDP-IP is left as an open question. Specifically, Rahwan et al. leave as an open question how the runtime of IDP-IP varies between the two extremes of $O(n^n)$ and $O(3^n)$ as the parameter m varies (Rahwan and Jennings, 2008a).

III.2.4 Heuristic Algorithms

Heuristic algorithms generally sacrifice quality guarantees for speed. This tradeoff is acceptable as long as, in practice, the algorithm is capable of returning good solutions quickly; however, it is usually not possible to verify the quality of a heuristically generated solution. For example, Sen and Dutta (Sen and Dutta, 2000) present a heuristic coalition structure generation algorithm based upon genetic algorithms (Eiben and Smith, 2003).

This dissertation focuses on algorithms for coalition structure generation that are either design-to-time or anytime. That is, this dissertation considers only algorithms that can provide quality guarantees on the solutions they generate.

III.3 Classes of Coalitional Games

This section surveys some results in both the complexity of computing different solution concepts and the complexity of coalition structure generation in compactly representable classes of coalitional games.

III.3.1 Complexity of Solution Concept Computation

The complexity of computing many different solution concepts in weighted voting games has been investigated (Elkind et al., 2007). The following complexity results are known:

1. Computing the Shapley value is #P-hard and even determining whether the Shapley value of an agent is 0 is NP-hard (Matsui and Matsui, 2001; Prasad and Kelly, 1990).

2. Determining if the core is empty can be done in polynomial time. The core of a weighted voting game is non-empty if and only if there is an agent that is present in

every winning coalition. Such an agent is referred to as a veto player. An imputation in the core can be constructed by giving each veto player an equal share of value of the grand coalition (Elkind et al., 2007).

3. Determining for a given ε if the ε-core of a given weighted voting game is non-empty is NP-hard. This result implies that determining $e(v)$ is NP-hard as well (Elkind et al., 2007).

4. Determining if a given imputation p is in the least-core is NP-hard. Likewise it is NP-hard to construct an imputation in the least-core of a given weighted voting game (Elkind et al., 2007).

5. Determining if the CS-core of a given weighted voting game is non-empty is NP-hard (Elkind et al., 2008).

6. Determining if a given coalition structure imputation pair (CS, p) is in the CS-core of a given weighted voting game is coNP-complete (Elkind et al., 2008).

However, the problems above involving the least core can be computed in pseudo-polynomial time[2] (Elkind et al., 2008, 2007).

Deng and Papadimitriou (Deng and Papadimitriou, 1994) study the complexity of computing a number of solution concepts in graph games. Deng and Papadimitriou show that in graph games, the Shapley value can be computed in polynomial time.

III.3.2 Complexity of Coalition Structure Generation

Aziz and Keijzer (Aziz and de Keijzer, 2011) show that an optimal coalition structure can be computed in polynomial time in a number of compactly representable coalitional games. While the coalition structure generation problem in weighted voting games is NP-complete,

[2]Recall that an algorithm runs in pseudo-polynomial time if it runs in time polynomial in the value of its input. In the case of weighted voting games, such an algorithm will run in time polynomial in the weights of the agents and the quota. Equivalently, such an algorithm will run in time polynomial in the size of its input if the weights and quota were given in unary.

Aziz and Keijzer present a polynomial time approximation algorithm that obtains a coalition structure with value at least $\frac{1}{2}$ that of the optimal. However, Aziz and Keijzer also show that the optimal coalition structure in weighted voting games cannot be approximated to a factor less than 2 in polynomial time, unless $P = NP$ (Aziz and de Keijzer, 2011).

Computing an optimal coalition structure in general graph games as well as on planar graph games is NP-complete (Voice et al., 2011). However, Voice et al. (Voice et al., 2011) show that an optimal coalition structure can be computed in polynomial time in some classes of minor free graphs.

Ohta et al. (Ohta et al., 2009) demonstrate that an optimal coalition structure can often be found quickly using constraint optimization techniques. In particular, Ohta et al.'s results apply to marginal contribution nets, a compact representation considered in Chapter VI.

III.4 Other Models of Coalition Formation

Much other research in coalition formation has focused on applications in task allocation (Abdallah and Lesser, 2004; Service, 2009; Service and Adams, 2011a; Shehory and Kraus, 1995, 1996, 1998; Vig and Adams, 2005, 2006; Vig, 2006; Vig and Adams, 2007). Typically, in such research the agents are assumed to be group rational and algorithms are developed to find coalitions of agents capable of solving a set of tasks with the greatest utility.

A common model of task based coalition formation assumes each agent has a fixed number of different resources. Each task requires some quantity of each resource to be successfully completed. A coalition of agents is capable of completing its assigned task only if for each resource required by the task, the sum of the quantities of that resource possessed by agents in the coalition is at least the quantity required by the task (Baliyarasimhuni and Beard, 2008; Service and Adams, 2011a; Shehory and Kraus, 1995, 1996, 1998; Tosic and Agha, 2005; Vig and Adams, 2006). Formally, each agent a and each task t has a resource vector r_a and $r_t \in \mathbb{R}^m$, where m is the number of resources. Each component of r_a and r_t

are non-negative. A coalition of agents C is capable of completing a task t if:

$$\sum_{a \in C} r_a \geq r_t,$$

where for $p, q \in \mathbb{R}^m$, $p \geq q$ if and only if every component of p is at least as great as its corresponding component of q. Each task is also assigned a utility that the agents collectively receive if the task is successfully completed. Shehory and Kraus (Shehory and Kraus, 1998) consider a slight variant of this model where the utility of a task is a function of the resources required by the task. This model assumes that resources are completely transferable between agents; an assumption that has been shown to not be valid in all domains (Vig and Adams, 2005).

There are several variants of the resource model. One important variant useful for multi-robot domains where agent resources are not necessarily transferable is known as the service model (Service and Adams, 2011a; Vig and Adams, 2005, 2006; Vig, 2006; Vig and Adams, 2007). Instead of resources, each task requires a set of services to be performed and each agent is capable of performing a set of different services. A coalition of agents is capable of completing a task if and only if it is possible to assign each agent in the coalition to perform one of the required services of the task, such that:

1. each agent is performing at most one service for the task,
2. each agent is capable of performing the service it is assigned to perform, and
3. all of the services required by the task are performed by some agent.

This dissertation focuses on which coalitions of agents will form and not on what those coalitions do once formed. As such, no notion of a task is assumed.

Wooldridge and Dunne (Wooldridge and Dunne, 2004) consider situations in which each agent has a set of goals. Agents attempt to form coalitions in order to achieve at least one of their goals (they are; however, indifferent as to which goal is achieved). Each

coalition of agents is capable of solving a number of different subsets of goals. The characteristic function in such qualitative coalitional games is of the form $v : 2^N \rightarrow 2^{2^G}$, where G is the set of all possible agent goals. They consider a succinct representation of the characteristic function using propositional logic. Wooldridge and Dunne consider the complexity of several computational problems associated with qualitative coalitional games. They consider questions such as:

1. Can a given coalition achieve at least one goal for each of its members (i.e., is the coalition successful)?

2. Given a coalition, is any subcoalition of this coalition successful?

3. Is the core of a given qualitative coalitional game empty?

Wooldridge and Dunne show that these questions, along with others, are all computationally intractable.

Wooldridge and Dunne (Wooldridge and Dunne, 2006) consider a situation similar to qualitative coalitional games, where agents are endowed with certain resources and goals require certain quantities of different resources to successfully be completed. In coalitional resource games, like qualitative coalitional games, each agent has a set of goals. A particular agent is satisfied if at least one of its goals is achieved; however, it is indifferent as to which goal is achieved. A coalition of agents can achieve a set of goals if and only if the collective resources of the agents in the coalition meet or exceed what is required by that set of goals. This model differs fundamentally from qualitative coalitional games where a characteristic function specifies which sets of goals are achievable. Wooldridge and Dunne (Wooldridge and Dunne, 2006) again consider several computational questions about coalitional resource games and show that many are intractable.

Both qualitative coalitional games and resource coalitional games are forms of non-transferable utility games. This dissertation considers only games with transferable utility.

Partition function games are another domain in which coalition structure generation has been studied (Michalak et al., 2008; Rahwan et al., 2009a). Partition function games are a generalization of the coalitional games considered by this dissertation. In a partition function game, the value of a coalition depends on both the members of the coalition as well as the coalitions that the other agents form (Michalak et al., 2008, 2009b; Rahwan et al., 2009a). That is, in partition function games there are externalities that affect the value of a coalition. This generalization significantly increases the computational difficulty of the coalition structure generation problem, as algorithms can no longer make use of the fact that the value of a coalition is independent of the coalition structure in which it appears. This dissertation does not consider this more general problem, as there is still a significant amount of improvement possible in the algorithms for coalition structure generation in the non-partition function form of coalitional games.

III.5 Related Problems

A number of standard problems in computer science are related to the coalition formation problem.

III.5.1 Set Partition

The weighted set partition problem is defined by a set of n elements $U = \{1, \cdots, n\}$, a collection of m sets $C = \{S_1, \cdots, S_m\}$, such that $\cup_{i=1}^{m} S_i = U$, and a weight function $w : C \rightarrow \mathbb{R}^+$ (Balas and Padberg, 1976). The set partition problem is to find a maximum weight subcollection of the sets $S_1, \cdots, S_m$ that form a partition of U. The coalition formation problem is naturally cast as an instance of the set partition problem, where U is the set of agents and C contains $2^n - 1$ sets, one for each coalition. Algorithms have been developed that can find an optimal set partition in $O^*(2^{n+b})$ time, where b is the number of bits required to represent the subset weights (Björklund et al., 2007, 2009).

III.5.2 Set Cover

The weighted set cover problem is defined by a set of n elements $U = \{1, \cdots, n\}$, a collection of m sets $C = \{S_1, \cdots, S_m\}$, such that $\cup_{i=1}^{m} S_i = U$, and a weight function $w : C \rightarrow \mathbb{R}^+$. The set cover problem is to find a minimum weight subcollection of the sets $S_1, \cdots, S_m$ whose union still contains U. Shehory and Kraus (Shehory and Kraus, 1995, 1998) have developed a coalition formation algorithm for task allocation based on an approximation algorithm for the set cover problem.

CHAPTER IV

Approximate Coalition Structure Generation

This chapter presents several approximation algorithms for the coalition structure generation problem. All of the presented algorithms extract approximate solutions from partially completed dynamic programming tables.

Two classes of approximation algorithms designed for arbitrary coalitional games are described in Sections IV.1 and IV.2. The presented algorithms are capable of generating better solution quality guarantees in lower worst case time than all currently existing algorithms. Section IV.1 presents a deterministic approximation algorithm, while Section IV.2 demonstrates how randomization can be employed to achieve better approximation ratios in less time.

Section IV.3 considers the coalition structure generation problem restricted to monotonic coalitional games. First, it is shown that any algorithm that guarantees better than a $\frac{1}{2}$-approximation to the coalition structure generation in monotonic games has running time $\Omega\left(\frac{2^n}{\sqrt{n}}\right)$. Second, an approximation algorithm for monotonic coalitional games is presented. To the best of the author's knowledge, this is the first algorithm specifically designed to leverage the monotonicity of the coalitional game for approximate coalition structure generation.

Many of the presented algorithms involve examining all coalitions of agents that contain at most some fixed fraction of the agents (i.e., all coalitions that consist of at most $\lfloor \frac{n}{r} \rfloor$ agents). The floor symbols are dropped from the statement of the algorithms and their correctness proofs in order to improve readability. For example, $\binom{n}{\frac{n}{r}}$ is written even though $\frac{n}{r}$ may not be an integer. The floor of $\frac{n}{r}$ is implicitly taken in these situations. Removal of the floor symbols affects only the readability of the formula and not the derived asymptotic bounds.

IV.1 Approximate Coalition Structure Generation in General Games

This section presents an algorithm for obtaining constant factor approximations of the optimal coalition structure in an arbitrary coalitional game (Service and Adams, 2010b).

A summary of various approximation ratios, and their corresponding running times, obtainable by the algorithm presented in this section are provided in Table IV.1. For example, the presented algorithm is capable of generating an approximate solution that is guaranteed to be within $66.\bar{6}\%$ of the optimal in $O^*(2.83^n)$ time. Prior to the algorithm presented in this section, the best approximation guarantee obtainable in $o^*(3^n)$ time by any known algorithm was $\frac{2}{n}$ (Sandholm et al., 1999; Rahwan et al., 2009b; Dang and Jennings, 2004).

Table IV.1: Approximation ratios and corresponding running times for the presented algorithm for arbitrary coalitional games. The first column provides the approximation ratio in terms of fraction of the optimal and the second column provides the corresponding runtime.

Approximation Ratio	Run Time
1	$O^*(3^n)$
$2/3$	$O^*(2.83^n)$
$1/2$	$O^*(2.59^n)$
$2/5$	$O^*(2.38^n)$
$1/3$	$O^*(2.22^n)$
$2/7$	$O^*(2.09^n)$
$1/4$	$O^*(2^n)$

Given a coalitional game (N, v), the presented algorithm obtains an approximate solution by applying one or two transformations to v, one after the other, to obtain new characteristic functions v' and v'' such that:

1. the values of the optimal solutions in v, v' and v'' are the same,
2. given a coalition structure, not necessarily optimal, in v' or v'', a coalition structure of equal or greater value can be constructed in v in polynomial time, and
3. v' and v'' both have optimal solutions consisting of a constant number, m, of coalitions.

Since v' has an optimal solution consisting of m coalitions, the highest valued coalition, C, in v' will have value at least $\frac{1}{m}$ times the value of an optimal solution to v', as C has value at least as great as all coalitions in any optimal solution to v'. Likewise, the sum of the values in v'' of two disjoint coalitions, C_1 and C_2 that maximize $v''(C_1)+v''(C_2)$ will be at least $\frac{2}{m}$ times the value of any optimal solution to v'', as C_1 and C_2 will have a combined value no less than the two highest valued coalitions in any optimal solution to v''. Algorithms are provided that extract the largest valued coalitions from v' and the two largest valued coalitions from v'' in $O^*(2^n)$ time.

The presented approximation technique employs the standard dynamic programming algorithm (Algorithm 1 on page 29) for coalition structure generation (Sandholm et al., 1999; Yeh, 1986). Recall that the dynamic programming algorithm is based on the observation that if C is a coalition in an optimal solution CS to a subproblem $N' \subseteq N$, then $CS \setminus \{C\}$ is an optimal solution to the subproblem $N' \setminus C$.

The proof of correctness of the presented approximation algorithm employs a generalization of the notion of superadditivity (Definition 6 on page 7) provided in Definition 36.

Definition 36 (k-supperadditive)**.** *A characteristic function v is* k-superadditive *iff for all $C, S \subseteq N$ such that*

1. $C \cap S = \emptyset$ *and*
2. $|C \cup S| \leq k$,

then $v(C)+v(S) \leq v(C \cup S)$.

Intuitively, v is k-superadditive if it is superadditive on all subproblems of k or fewer agents. As with superadditive games where there is an optimal solution that consists of exactly one coalition, when it is known that a game is $\frac{n}{r}$-superadditive, an upper bound can be placed on the number of coalitions in some optimal solution. For example, if it is known that v is $\frac{n}{2}$-superadditive, then there must be an optimal coalition structure that consists of at most 3 coalitions. A general result regarding the number of coalitions in

an optimal coalition structure in k-superadditive games is presented in Theorem 4. It is then shown how to transform an arbitrary coalitional game into a k-superadditive game and subsequently how to extract an approximation solution. Theorem 4 generalizes the well known fact that, in superadditive games, the coalition structure containing only the grand coalition is optimal.

Theorem 4. *If v is $\frac{n}{r}$-superadditive, then v has an optimal solution that consists of at most $2r-1$ coalitions.*

Proof. Let CS be an optimal coalition structures with the least number of coalitions. Assume that $|CS| \geq 2r$. Let C_1 and C_2 be the two smallest coalitions in CS. Since CS contains at least $2r$ coalitions, $|C_1| + |C_2| \leq 2 \cdot \frac{n}{2r} = \frac{n}{r}$. Since v is $\frac{n}{r}$-supperadditive, $v(C_1 \cup C_2) \geq v(C_1) + v(C_2)$. This is a contradiction since the coalition structure

$$CS' = (CS \setminus \{C_1, C_2\}) \cup \{C_1 \cup C_2\}$$

is an optimal coalition structure with fewer coalitions. □

The following is an immediate corollary of Theorem 4:

Corollary 5. *If v is $\frac{2n}{2r-1}$-superadditive, then v has an optimal solution that consists of $2r-2$ coalitions.*

Proof. Since $\frac{2n}{2r-1} > \frac{n}{r}$, v is also $\frac{n}{r}$-superadditive and by Theorem 4 has an optimal solution that consists of at most $2r-1$ coalitions. If v contains an optimal solution on $2r-2$ or fewer coalitions, then the proof is complete. Thus, assume the smallest optimal solution, CS in v consists of exactly $2r-1$ coalitions.

Let C_1 and C_2 be the smallest two coalitions in CS and assume that $|C_1 \cup C_2| > \frac{2n}{2r-1}$. Thus, at least one of C_1 or C_2 must contain more than $\frac{n}{2r-1}$ agents. Since C_1 and C_2 are the two smallest coalitions in CS, all remaining coalitions must also contain more than $\frac{n}{2r-1}$

agents. As there are $2r-3$ coalitions in CS other than C_1 and C_2, the total number of agents is bounded below by:

$$n > (2r-3) \cdot \frac{n}{2r-1} + \frac{2n}{2r-1} = n,$$

a contradiction. Thus $|C_1 \cup C_2| \leq \frac{2n}{2r-1}$. Define CS' as:

$$CS' = CS \cup \{C_1 \cup C_2\} \setminus \{C_1, C_2\}.$$

Since v is $\frac{2n}{2r-1}$-superadditive, $v(C_1 \cup C_2) \geq v(C_1) + v(C_2)$. Thus, $v(CS') = v(CS)$ (since CS was optimal). However, CS' contains one less coalition than CS, contradicting the assumption that the smallest optimal solution in v consisted of $2r-1$ coalitions. □

Recall that $opt(C)$ is the value of the optimal solution to the subproblem consisting of only those agents in C and that $sol(C)$ is the optimal coalition structure over the agents in C.

Definition 37 ($v_{\frac{k}{r}}$)**.** *Let k and r be positive integers. Define* $v_{\frac{k}{r}}$ *as:*

$$v_{\frac{k}{r}}(C) = \begin{cases} opt(C) & \text{if } |C| \leq \frac{kn}{r} \\ v(C) & \text{otherwise.} \end{cases}$$

Theorem 6 is an immediate observation.

Theorem 6. $v_{\frac{k}{r}}$ *is* $\frac{kn}{r}$*-superadditive.*

Proof. The theorem is clearly true, since for any $C, S \subseteq N$ such that $C \cap S = \emptyset$ and $|C \cup S| \leq \frac{kn}{r}$, $sol(C) \cup sol(S)$ is a coalition structure over $C \cup S$ with value $v_{\frac{k}{r}}(C) + v_{\frac{k}{r}}(S)$. □

Algorithm 2 provides pseudo-code for the construction of $v_{\frac{k}{r}}$. Intuitively, Algorithm 2 runs the dynamic programming algorithm for only those coalitions consisting of $\frac{kn}{r}$ or fewer agents. For clarity, the code for maintaining pointers from each subproblem N' to subproblems C and $N' \setminus C$ for constructing an optimal solution is not provided.

Algorithm 2 Constructing $v_{\frac{k}{r}}$

for $i = 1$ to $\frac{kn}{r}$ **do**
  for $C \subseteq N, |C| = i$ **do**
    $v_{\frac{k}{r}}(C) \leftarrow v(C)$
    for $C' \subseteq C$ **do**
      if $v_{\frac{k}{r}}(C') + v_{\frac{k}{r}}(C \setminus C') > v_{\frac{k}{r}}(C)$ **then**
        $v_{\frac{k}{r}}(C) \leftarrow v_{\frac{k}{r}}(C') + v_{\frac{k}{r}}(C \setminus C')$
      end if
    end for
  end for
end for
$v_{\frac{k}{r}}(C) = v(C)$ for all C, such that $|C| > \frac{kn}{r}$

Note that every optimal solution to v is also an optimal solution to $v_{\frac{k}{r}}$, since if C is in an optimal solution to v, then $opt_v(C) = v(C)$. Given a solution $CS_{v_{\frac{k}{r}}}$ to $v_{\frac{k}{r}}$, a solution CS_v to v of equal value can be constructed as follows, assuming CS_v is initially empty:

1. for $C \in CS_{v_{\frac{k}{r}}}$ such that $|C| > \frac{k}{r}$, let $CS_v \leftarrow CS_v \cup \{C\}$, and
2. for $C \in CS_{v_{\frac{k}{r}}}$ such that $|C| \leq \frac{k}{r}$, let $CS_v \leftarrow CS_v \cup sol_{v_{\frac{k}{r}}}(C)$.

Lemma 1 is used to determine the runtime of the various approximation algorithms.

Lemma 1. *Let r be a fixed integer, then* $\binom{n}{n/r} = O^*\left(\frac{r^n}{(r-1)^{\frac{(r-1)n}{r}}}\right)$.

Proof. Recall that Stirling's approximation states:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n (1 + o(1)) = \Theta\left(\sqrt{n}\left(\frac{n}{e}\right)^n\right).$$

$$\begin{aligned}
\binom{n}{n/r} &= \frac{n!}{\frac{n}{r}!\frac{(r-1)n}{r}!} \\
&= O(1)\cdot\left(\frac{\sqrt{n}(\frac{n}{e})^n}{\sqrt{\frac{n}{r}}\left(\frac{(\frac{n}{r})}{e}\right)^{\frac{n}{r}}\cdot\sqrt{n-\frac{n}{r}}\left(\frac{(n-\frac{n}{r})}{e}\right)^{(n-\frac{n}{r})}}\right) \\
&= O(1)\cdot\left(\frac{n^n}{\sqrt{n}(\frac{n}{r})^{\frac{n}{r}}\cdot(n-\frac{n}{r})^{(n-\frac{n}{r})}}\right) \\
&= O\left(\frac{r^n}{\sqrt{n}(r-1)^{\frac{(r-1)n}{r}}}\right) \\
&= O^*\left(\frac{r^n}{(r-1)^{\frac{(r-1)n}{r}}}\right).
\end{aligned}$$

□

The runtime of Algorithm 2 is provided in Theorem 7

Theorem 7. *Algorithm 2 computes $v_{\frac{k}{r}}$, for $k \leq \frac{r}{2}$ and $r \geq 2$, in:*

$$O^*\left(\frac{r^n 2^{\frac{kn}{r}}}{k^{\frac{kn}{r}}(r-k)^{\frac{(r-k)n}{r}}}\right).$$

Proof. Algorithm 2 determines the optimal solution to each subproblem of N consisting of $\frac{kn}{r}$ or fewer agents. For a subproblem of size n', $O^*(2^{n'})$ time is required. Ignoring polynomial factors, the total runtime is bounded above by:

$$\sum_{0<i\leq\frac{kn}{r}}\binom{n}{i}2^i \leq \frac{kn}{r}\binom{n}{\frac{kn}{r}}2^{\frac{kn}{r}} = O^*\left(\frac{r^n 2^{\frac{kn}{r}}}{k^{\frac{kn}{r}}(r-k)^{\frac{(r-k)n}{r}}}\right),$$

by Lemma 1.

□

After $v_{\frac{1}{r}}$ is constructed, the highest valued coalition can be extracted in $O^*(2^n)$ time by looping through all subsets of N. Since $v_{\frac{1}{r}}$ has an optimal solution consisting of $2r-1$ or

fewer coalitions, the highest valued coalition is at least $\frac{1}{2r-1}$ times the value of the optimal solution. Hence, a $\frac{1}{8}$-approximate solution can be obtained in $O^*(2^n)$ time by constructing $v_{\frac{2}{9}}$ and then extracting the highest valued coalition.

It is now shown how to extract two coalitions, C_1 and C_2, from an arbitrary v that maximize $v(C_1)+v(C_2)$ in $O^*(2^n)$.

Definition 38 (v_{max})**.** *Define v_{max} as:*

$$v_{max}(C) = \max_{S \subseteq C} v(S).$$

Intuitively, $v_{max}(C)$ represents the highest valued coalition that consists of only members in C. Algorithm 3 provides pseudo-code for constructing v_{max} given v. Algorithm 3 is based on the observation that

$$v_{max}(C) = \max_{S \subseteq C} v(S) = \max\{v(C), \max_{a \in C}\{v_{max}(C \setminus \{a\})\}.$$

Since Algorithm 3 constructs v_{max} in the order of increasing coalition size, when computing $v_{max}(C)$, Algorithm 3 takes the maximum over $|C|+1$ values (i.e., $v(C)$ and $v_{max}(C \setminus \{a\})$ for each $a \in C$).

Algorithm 3 Constructing v_{max}

```
1: for k = 1 to n do
2:   for C ⊆ N, |C| = k do
3:     v_max(C) ← v(C)
4:     for a ∈ C do
5:       if v_max(C \ {a}) > v_max(C) then
6:         v_max(C) ← v_max(C \ {a})
7:       end if
8:     end for
9:   end for
10: end for
```

During the construction of v_{max}, for each $S \subseteq N$, in addition to storing the value of the largest subset of S in $v_{max}(S)$, a pointer from S to $\text{argmax}_{C \subseteq S} v(C)$ is stored as well. For

clarity pseudo-code for storing such pointers is omitted from Algorithm 3.

Given a solution $CS_{v_{max}}$ to v_{max}, a solution CS_v to v of equal or greater value can be constructed as follows. Assume CS_v to be initially empty,

$$\text{for } C \in CS_{v_{max}}, \text{ let } CS_v \leftarrow CS_v \cup \{\operatorname*{argmax}_{S \subseteq C} v(S)\}.$$

Note that this procedure may not result in a partitioning of the agents (i.e., some agents may not appear in any of the coalitions in CS_v). In this case, CS_v can be extended to a partitioning of N by adding to CS_v the coalition C containing all agents that do not appear in any coalition in CS_v.

The runtime of Algorithm 3 is provided in Theorem 8.

Theorem 8. *Algorithm 3 computes v_{max} in $O^*(2^n)$ time.*

Proof. For each set C of k agents, Algorithm 3 determines the maximum value over the previously determined entries of all k subsets of C of $k-1$ agents and compares it to $v(C)$. There are $\binom{n}{k}$ subsets of n agents of size k and for each, the maximum must be taken over $k+1$ values. The resulting run time is bounded above by:

$$\begin{aligned}
\sum_{0<k\leq n} \binom{n}{k}(k+1) &= \sum_{0<k\leq n} \binom{n}{k}k + \sum_{0<k\leq n} \binom{n}{k} \\
&= n2^{n-1} + 2^n - 1, \\
&= O^*(2^n).
\end{aligned}$$

□

Algorithm 4 provides pseudo-code for extracting two disjoint coalitions, C_1 and C_2, from N that maximize $v_{max}(C_1) + v_{max}(C_2)$. Algorithm 4 takes the maximum of $v_{max}(C) + v_{max}(N \setminus C)$ for $C \subseteq N$. By the definition of v_{max}, Algorithm 4 clearly results in an optimal

partitioning of N into two sets. Since Algorithm 4 computes

$$\operatorname*{argmax}_{C_1 \cap C_2 = \emptyset} (v_{max}(C_1) + v_{max}(C_2))$$

by a single loop through all subsets of N, it runs in $O^*(2^n)$ time.

Algorithm 4 Computing $\operatorname*{argmax}_{C_1 \cap C_2 = \emptyset} (v_{max}(C_1) + v_{max}(C_2))$

```
1: C_1 = N
2: C_2 = ∅
3: for C ⊆ N do
4:     if v_max(C) + v_max(N \ C) > v_max(C_1) + v_max(C_2) then
5:         C_1 = C
6:         C_2 = N \ C
7:     end if
8: end for
```

Given v, the overall approximation technique works as follows:

1. construct $v_{\frac{k}{r}}$ from v, using Algorithm 2,
2. construct v_{max} from $v_{\frac{k}{r}}$, using Algorithm 3,
3. extract the optimal partitioning of N into two sets from v_{max}, using Algorithm 4,
4. convert the extracted solution to v_{max} to a solution to $v_{\frac{k}{r}}$ and then to a solution to v.

The runtime of this procedure is the maximum of $O^*\left(\frac{r^n 2^{\frac{kn}{r}}}{k^{\frac{kn}{r}}(r-k)^{\frac{(r-k)n}{r}}}\right)$ and $O^*(2^n)$, the times required to compute $v_{\frac{k}{r}}$ and v_{max}, respectively.

For example, given a coalitional game v, an approximate solution that is within $\frac{2}{3}$ of the optimal can be obtained in $O^*(2.83^n)$ time by first constructing $v_{\frac{1}{2}}$ from v, then constructing v_{max} from $v_{\frac{1}{2}}$ and finally extracting the optimal partitioning of N into two sets from v_{max} and converting that to a coalition structure in v of equal or greater value. Similarly, a $\frac{1}{2}$-approximate solution can be obtained in $O^*(2.59)$ time by constructing $v_{\frac{2}{5}}$ from v, then constructing v_{max} from $v_{\frac{2}{5}}$ and finally extracting the optimal partitioning of N into two sets from v_{max} and converting that to a solution to v.

IV.2 Randomized Coalition Structure Generation

This section demonstrates how randomization can be employed to achieve better approximation guarantees in less time (Service and Adams, 2011c). The algorithms presented in this section all succeed in finding an f-approximate solution with probability greater than $\frac{e-1}{e}$. Rerunning the algorithm, for example, three times and selecting the best solution found over all three runs, results in an f-approximate solution with probability at least $1-(1-\frac{e-1}{e})^3 > 0.95$ (i.e., the probability that all three runs fail to find an f-approximate solution is no more than $(1-\frac{e-1}{e})^3$). However, the performance of the algorithm may not be the only probabilistic element. The algorithm's runtime may also be a random variable. When an algorithm's runtime is a random variable, the algorithm's *expected* runtime is reported.

All of the presented results follow from a careful analysis of the number of coalitions, and their respective sizes, for a particular optimal solution. As a result, a single randomized approximate algorithm that is capable of generating all the approximation ratios is not presented. Instead, it is necessary to tailor each algorithm to each approximation ratio. Therefore, a different algorithm for each approximation ratio is presented.

All the presented randomized approximation algorithms employ the same approach as the deterministic approximation algorithm. Given an arbitrary coalitional game v, first construct $v_{\frac{k}{r}}$, for appropriately chosen k and r. Second, extract m disjoint coalitions that maximize their combined value. Section IV.1 showed how to extract 2 disjoint coalitions that maximize their combined value in $O^*(2^n)$ time. The main contribution of this section is to show that in some situations, extracting m disjoint, possible empty coalitions that maximize the sum of their values for $m = 3,4$ is possible using simple randomized algorithms.

In summary, the presented randomized approximation techniques are variations of the following procedure:

1. Construct $v_{\frac{k}{r}}$ to ensure that there exists an optimal coalition structure that contains only m coalitions (for some constant m).

2. Run, possibly multiple solution extraction procedures and return the best solution.

Multiple solution extraction procedures are required because the smallest optimal solution may contain $1, \cdots, m$ coalitions. Each extraction procedure is designed to handle some subset of the possible sizes of the smallest optimal solution.

Recall that constructing $v_{\frac{k}{r}}$ requires $O^*\left(\frac{r^n 2^{\frac{kn}{r}}}{k^{\frac{kn}{r}}(r-k)^{\frac{(r-k)n}{r}}}\right)$ time. All but one of the presented algorithms will require $v_{\frac{k}{r}}$ to be monotonic. Recall that enforcing the constraint that $v_{\frac{k}{r}}$ be monotonic requires an additional $O^*(2^n)$ time. Unless otherwise stated, it is assumed that $v_{\frac{k}{r}}$ is monotonic.

Algorithm 4 shows how to extract two disjoint coalitions C_1 and C_2 that maximize $v_{\frac{k}{r}}(C_1) + v_{\frac{k}{r}}(C_2)$. The second and third extraction procedures (Algorithm 5 and Algorithm 6) are randomized and require a-priori knowledge of the maximum sizes of the coalitions to be extracted.

Algorithm 5 works as follows: Assume it is known that there are three disjoint coalitions C_1^*, C_2^* and C_3^* with total value at least f times that of an optimal solution. A simple randomized approach to find three disjoint coalitions with total value at least f times that of the optimal simply adds each agent to one of three sets with equal probability. That is, with probability $\frac{1}{3}$, each agent a_i is added to coalition C_1, with probability $\frac{1}{3}$ a_i is added to C_2, and with probability $\frac{1}{3}$ a_i is added to C_3. Since v_{max} is monotonic, as long as $C_i^* \subseteq C_i$, for $i = 1,2,3$, then the total value of C_1, C_2 and C_3 will be at least f times that of an optimal solution. The probability that $C_i^* \subseteq C_i$, for $i = 1,2,3$ is simply $(\frac{1}{3})^{|C_1^*|+|C_2^*|+|C_3^*|}$. Lemma 3 shows that in order to be guaranteed to find C_1, C_2 and C_3 with high probability, it is sufficient to repeat this procedure $O(3^{|C_1^*|+|C_2^*|+|C_3^*|})$ times.

The proof of Lemma 3 will require the following result.

Lemma 2. *In a binomial distribution with success probability p after $\lceil \frac{1}{p} \rceil$ trials, the probability of at least one success is at least $\frac{e-1}{e}$ (approximately, 63.21%).*

Proof. Since the probability that any single trial is unsuccessful is $1-p$, the probability

Algorithm 5 Extracting three coalitions from a monotonic v. Requires one parameter: *num_iterations*.

```
1: C_1 ← ∅
2: C_2 ← ∅
3: C_3 ← ∅
4: for i = 0 to num_iterations do
5:    C'_1 ← ∅
6:    C'_2 ← ∅
7:    C'_3 ← ∅
8:    for k = 0 to num_agents do
9:       Add agent a_k to one of the coalitions C'_1, C'_2 or C'_3 with equal probability.
10:   end for
11:   if v(C'_1) + v(C'_2) + v(C'_3) > v(C_1) + v(C_2) + v(C_3) then
12:      C_1 ← C'_1
13:      C_2 ← C'_2
14:      C_3 ← C'_3
15:   end if
16: end for
```

that n trials are all unsuccessful is $(1-p)^n$. Hence, the probability of at least one successful trial in n trials is $1-(1-p)^n$.

Consider the case were $n = \lceil \frac{1}{p} \rceil$, then

$$(1-p)^n \leq \left(1 - \frac{1}{n}\right)^n.$$

Since $(1-\frac{1}{n})^n$ increases monotonically to $\frac{1}{e}$ as $n \to \infty$, then $(1-\frac{1}{n})^n \leq \frac{1}{e}$. Hence, $1-(1-p)^{\frac{1}{p}} \geq \frac{e-1}{e}$. □

Lemma 3 provides a proof of the correctness and runtime of Algorithm 5.

Lemma 3. *Let C_1^*, C_2^* and C_3^* be three disjoint coalitions from a set of n agents. If v is monotonic, then after $O(3^{|C_1^*|+|C_2^*|+|C_3^*|})$ iterations the best solution found by Algorithm 5 has value of at least $v(C_1^*)+v(C_2^*)+v(C_3^*)$, with probability at least $\frac{e-1}{e}$.*

Proof. During a single iteration of Algorithm 5, the probability that all agents in C_i^* are placed in C_i', for $i = 1, 2$, and 3, is simply $\left(\frac{1}{3}\right)^{|C_1^*|+|C_2^*|+|C_3^*|}$. The iterations of Algorithm 5 form a binomial distribution with success probability $\left(\frac{1}{3}\right)^{|C_1^*|+|C_2^*|+|C_3^*|}$. Therefore, after

$3^{|C_1^*|+|C_2^*|+|C_3^*|}$ iterations, with probability greater than $\frac{e-1}{e}$, Algorithm 5 succeeds in finding three coalitions $C_1^* \subseteq C_1'$, $C_2^* \subseteq C_2'$ and $C_3^* \subseteq C_3'$. Since v is monotonic, $v(C_1') + v(C_2') + v(C_3') \geq v(C_1^*) + v(C_2^*) + v(C_3^*)$. □

The following variation of Algorithm 5 (henceforth referred to as Algorithm 5(a)) was found to be more efficient in practice. Rather then place each agent in C_1', C_2' or C_3' with equal probability, each agent a_i is placed in C_1' with probability 0.375 and in C_2' and C_3' with probability 0.3125. The increased efficiency stems from the fact that the probabilities 0.375 and 0.3125 can be created easily from a fair coin, while generating an event with $\frac{1}{3}$ probability cannot. All of our empirical results employ Algorithm 5(a). The use of Algorithm 5(a) does not affect the presented asymptotic results. Lemma 4 shows that the required number of iterations increases only slightly under in Algoritm 5(a).

Lemma 4. *Let C_1^*, C_2^* and C_3^* be three disjoint coalitions from a set of n agents. If v is monotonic, then after $O(3.012^{|C_1^*|+|C_2^*|+|C_3^*|})$ iterations the best solution found by Algorithm 5(a) has value at least $v(C_1^*) + v(C_2^*) + v(C_3^*)$ with probability at least $\frac{e-1}{e}$.*

Proof. During a single iteration of Algorithm 5(a), the probability that all agents in C_i^* are placed in C_i', for $i = 1, 2$, and 3, is simply $0.375^{|C_1^*|} \cdot 0.3125^{|C_2^*|+|C_3^*|}$. The iterations of Algorithm 5(a) form a binomial distribution with success probability $0.375^{|C_1^*|} \cdot 0.3125^{|C_2^*|+|C_3^*|}$. Therefore, after $\left(\frac{1}{0.375}\right)^{|C_1^*|} \cdot \left(\frac{1}{0.3125}\right)^{|C_2^*|+|C_3^*|}$ iterations, with probability greater than $\frac{e-1}{e}$, Algorithm 5(a) succeeds in finding three coalitions $C_1^* \subseteq C_1'$, $C_2^* \subseteq C_2'$ and $C_3^* \subseteq C_3'$. Since v is monotonic: $v(C_1') + v(C_2') + v(C_3') \geq v(C_1^*) + v(C_2^*) + v(C_3^*)$.

Assume without loss of generality that $|C_1| \geq |C_2| \geq |C_3|$. Thus, $|C_2| + |C_3| \leq \left(\frac{2}{3}\right)(|C_1| + |C_2| + |C_3|)$. Therefore, as $\frac{1}{0.375} < \frac{1}{0.3125}$ the total number of iterations required by the variation on Algorithm 5(a) is at most:

$$\left(\frac{1}{0.375}\right)^{\frac{|C_1|+|C_2|+|C_3|}{3}} \cdot \left(\frac{1}{0.3125}\right)^{\frac{2(|C_1|+|C_2|+|C_3|)}{3}} = O(3.012^{(|C_1|+|C_2|+|C_3|)}).$$

□

The new randomized approximation algorithms can be presented given the provided Lemmas. Table IV.2 presents a summary of the approximate ratios and running times obtainable by the new randomized algorithms.

Table IV.2: Approximation ratios and corresponding running times for the presented algorithm for arbitrary coalitional games. The first column provides the approximate ratio of the algorithm in terms of the fraction of the value of the optimal solution guaranteed. The second and third columns list the time required to obtain the stated approximation ratio for the deterministic and randomized algorithms, respectively.

Approximation Ratio	Deterministic Alg. Runtime	Randomized Alg. Runtime
$2/3$	$O^*(2.83^n)$	$O^*(2.587^n)$
$3/5$	N/A	$O^*(2.382^n)$
$1/2$	$O^*(2.59^n)$	$O^*(2.09^n)$
$1/3$	$O^*(2.22^n)$	$O^*(2^n)$

Theorem 9. *A $\frac{2}{3}$-approximate coalition structure in v can be found in $O^*(2.587^n)$ time.*

Proof. A $\frac{2}{3}$-approximation can be obtained as follows: First construct $v_{\frac{2}{5}}$ in $O^*(2.587^n)$ time, then run Algorithm 4 to find two disjoint coalitions that maximize the sum of their values. Finally run Algorithm 5 for $3^{\frac{4n}{5}} = O(2.41^n)$ iterations. This procedure guarantees a $\frac{2}{3}$-approximate solution with high probability.

Let $CS^* = \{C_1, \cdots, C_k\}$ be an optimal coalition structure in $v_{\frac{2}{5}}$ with the least number of coalitions. By Corollary 5, $|CS^*| \leq 4$. Without loss of generality, assume that

$$|C_1| \geq |C_2| \geq |C_3| \geq |C_4|.$$

There are two cases to consider.

1. $|CS^*| \leq 3$. Algorithm 4 returns a $\frac{2}{3}$-approximate solution in $O(2^n)$ time.

2. $|CS^*| = 4$. Since $v_{\frac{2}{5}}$ is $\frac{2n}{5}$-superadditive, $|C_3| + |C_4| > \frac{2n}{5}$. Otherwise $\{C_1, C_2, C_3 \cup C_4\}$ is an optimal coalition structure with fewer coalitions. Since $|C_3| \geq |C_4|$, it must be the case that $|C_3| > \frac{n}{5}$. Therefore, $|C_1|, |C_2| > \frac{n}{5}$.

At least one of the following must be true:

(a) $v(C_1) + v(C_2) \geq \frac{2}{3}v(CS^*)$,

(b) $v(C_1) + v(C_3) + v(C_4) \geq \frac{2}{3}v(CS^*)$ or

(c) $v(C_2) + v(C_3) + v(C_4) \geq \frac{2}{3}v(CS^*)$

since, if all three inequalities were false, summing the right and left hand sides yields:

$$2(v(C_1) + v(C_2) + v(C_3) + v(C_4)) < 2v(CS^*).$$

However, this is a contradiction as $v(C_1) + v(C_2) + v(C_3) + v(C_4) = v(CS^*)$.

If $v(C_1) + v(C_2) \geq \frac{2}{3}v(CS^*)$, then Algorithm 4 returns a $\frac{2}{3}$-approximate solution in $O^*(2^n)$ time. If one of the second or third cases is true, then both $|C_1| + |C_3| + |C_4|$ and $|C_2| + |C_3| + |C_4|$ contain fewer than $\frac{4n}{5}$ coalitions. Therefore, by Lemma 3, Algorithm 5 returns a $\frac{2}{3}$-approximate solution in $O(3^{\frac{4n}{5}}) = O(2.41^n)$ iterations.

The bottleneck step is constructing $v_{\frac{2}{5}}$, which requires $O^*(2.587^n)$ time. □

Theorem 10. *A $\frac{3}{5}$-approximate coalition structure in v can be found in $O^*(2.38^n)$ time.*

Proof. First construct $v_{\frac{1}{3}}$ in $O^*(2.38^n)$ time. After constructing $v_{\frac{1}{3}}$, run Algorithm 4 followed by Algorithm 5 for $3^{\frac{3n}{4}} = O(2.28^n)$ iterations.

Let $CS^* = \{C_1, \cdots, C_k\}$ be an optimal coalition structure in $v_{\frac{1}{3}}$ with the least number of coalitions. By Theorem 4, $|CS^*| \leq 5$. Without loss of generality, assume that

$$|C_1| \geq |C_2| \geq |C_3| \geq |C_4| \geq |C_5|.$$

There are three cases to consider.

1. $|CS^*| \leq 3$. Algorithm 4 will return a $\frac{2}{3} > \frac{3}{5}$-approximate solution in $O^*(2^n)$ time.

2. $|CS^*| = 4$. If $v(C_2) + v(C_3) + v(C_4) < \frac{3}{5}v(CS^*)$, then $v(C_1) > \frac{2}{5}v(CS^*)$. Let C_i be the coalition in $\{C_2, C_3, C_4\}$ with the highest value. Since $v(C_2) + v(C_3) + v(C_4) = v(CS^*) - v(C_1)$, then $v(C_i) \geq \frac{1}{3}(v(CS^*) - v(C_1))$. Therefore,

$$\begin{aligned} v(C_i) + v(C_1) &\geq \frac{v(CS^*)}{3} + \frac{2}{3}v(C_1) \\ &> \frac{v(CS^*)}{3} + \frac{4}{15}v(CS^*) \\ &= \frac{9}{15}v(CS^*) \\ &= \frac{3}{5}v(CS^*). \end{aligned}$$

Hence, either $v(C_2) + v(C_3) + v(C_4) \geq \frac{3}{5}v(CS^*)$ or $v(C_1) + v(C_i) \geq \frac{3}{5}v(CS^*)$. If $v(C_2) + v(C_3) + v(C_4) \geq \frac{3}{5}v(CS^*)$, then Algorithm 5 returns a $\frac{3}{5}$-approximate solution in $O^*(3^{\frac{3n}{4}}) = O^*(2.28^n)$ time because $|C_1| \geq \frac{n}{4}$. If $v(C_1) + v(C_i) \geq \frac{3}{5}v(CS^*)$ then Algorithm 4 returns a $\frac{3}{5}$-approximate solution.

3. $|CS^*| = 5$. Since v is $\frac{n}{3}$-superadditive, any three coalitions C_i, C_j and C_k can contain at most $\frac{2n}{3}$ agents. Since the three highest valued coalitions in CS^* must be a $\frac{3}{5}$-approximate solution, then Algorithm 5 returns a $\frac{3}{5}$-approximate solution in $O(3^{\frac{2n}{3}}) = O^*(2.09^n)$ iterations.

□

Theorem 11. *A $\frac{1}{3}$-approximate coalition structure in v can be found in $O^*(2^n)$ time.*

Proof. First construct $v_{\frac{2}{9}}$ in $O^*(2^n)$ time. Run Algorithm 4 to identify two disjoint coalitions that maximize the sum of their values, followed by Algorithm 5 for $3^{\frac{5n}{12}} = O(1.59^n)$ iterations.

This procedure guarantees a $\frac{1}{3}$-approximate solution with high probability.

Let $CS^* = \{C_1, \cdots, C_k\}$ be an optimal coalition structure in $v_{\frac{2}{9}}$ with the least number of

coalitions. By Corollary 5, $|CS^*| \leq 8$. Without loss of generality, assume that

$$|C_1| \geq |C_2| \geq \cdots \geq |C_k|.$$

There are three cases to consider.

1. $|CS^*| \leq 6$. Algorithm 4 returns a $\frac{1}{3}$-approximate solution in $O^*(2^n)$ time.

2. $|CS^*| = 7$. One of $\{C_1, C_2\}$, $\{C_3, C_4\}$, and $\{C_5, C_6, C_7\}$ must have total value at least $\frac{v(CS^*)}{3}$. If $\{C_1, C_2\}$ or $\{C_3, C_4\}$ has total value at least $\frac{v(CS^*)}{3}$, then Algorithm 4 returns a $\frac{1}{3}$-approximation solution. Otherwise, Algorithm 5 returns a $\frac{1}{3}$-approximation solution in $O(3^{\frac{3n}{7}}) = O(1.61^n)$ iterations, since $|C_5| + |C_6| + |C_7| \leq \frac{3n}{7}$.

3. $|CS^*| = 8$. One of $\{C_1, C_2\}$, $\{C_3, C_4, C_5\}$, and $\{C_6, C_7, C_8\}$ must have total value at least $\frac{v(CS^*)}{3}$. If $\{C_1, C_2\}$ has value at least $\frac{v(CS^*)}{3}$, then Algorithm 4 will return a $\frac{1}{3}$-approximate solution.

   Notice that $|C_1| + |C_2| \geq \frac{2n}{8} = \frac{n}{4}$ and, since $v_{\frac{2}{9}}$ is $\frac{2n}{9}$-superadditive, $|C_7| + |C_8| > \frac{2n}{9}$. Therefore, $|C_6|, |C_7| > \frac{n}{9}$ and hence $|C_6| + |C_7| + |C_8| \geq \frac{n}{3}$. Thus, $n - \frac{n}{4} - \frac{n}{3} = \frac{5n}{12} \geq |C_3| + |C_4| + |C_5| \geq |C_6 + |C_7| + |C_8|$. Therefore, if either $\{C_3, C_4, C_5\}$ or $\{C_6, C_7, C_8\}$ has total value at least $\frac{v(CS^*)}{3}$, then Algorithm 5 will find a $\frac{1}{3}$-approximation in $O(3^{\frac{5n}{12}}) = O(1.59^n)$ iterations.

□

The $\frac{1}{2}$-approximation algorithm requires an additional solution extraction procedure that works as follows. Let v be a monotonic game and assume that it is known that there are four disjoint coalitions C_1^*, C_2^*, C_3^* and C_4^* with total value at least f times the value of an optimal coalition structure. Also assume that $|C_1^*| + |C_3^*|, |C_2^*| + |C_4^*| \leq \frac{n}{3}$ and $|C_3^*|, |C_4^*| \leq u$ for some sufficiently small constant u.

The fourth solution extraction procedure begins by placing each agent randomly in one of two sets, S_1 or S_2, with equal probability. After randomly distributing the agents

among S_1 and S_2, the coalition $C_{S_1} \subset S_1$ that maximizes $v(C_{S_1}) + v(S_1 \setminus C_{S_1})$ subject to $|C_{S_1}| = u$ is found. Notice that, since v is monotonic, this is equivalent to finding two coalitions $C_1, C_2 \subset S_1$ that maximize $v(C_1) + v(C_2)$ subject to at least one of the coalitions containing u agents. This same process is repeated for S_2. That is, the coalition $C_{S_2} \subset S_2$ that maximizes $v(C_{S_2}) + v(S_2 \setminus C_{S_2})$ subject to $|C_{S_2}| = u$ is found. The maximum over all iterations of $v(C_{S_1}) + v(S_1 \setminus C_{S_1}) + v(C_{S_2}) + v(S_2 \setminus C_{S_2})$ is returned.

Algorithm 6 Extracting four coalitions from a monotonic v. Requires two parameters: *num_iterations* and *upperbound*.

```
C_i ← ∅ for i = 1,2,3,4
for i = 0 to num_iterations do
    C'_i ← ∅ for i = 1,2,3,4
    for k = 0 to num_agents do
        Add agent a_k to one of S_1 and S_2 with equal probability
    end for
    for A ⊂ S_1, |A| = upperbound do
        if v(A) + v(S_1 \ A) > v(C_1) + v(C_3) then
            C'_1 ← S_1 \ A
            C'_3 ← A
        end if
    end for
    for A ⊂ S_2, |A| = upperbound do
        if v(A) + v(S_2 \ A) > v(C_2) + v(C_4) then
            C'_2 ← S_2 \ A
            C'_4 ← A
        end if
    end for
    if v(C'_1) + v(C'_2) + v(C'_3) + v(C'_4) > v(C_1) + v(C_2) + v(C_3) + v(C_4) then
        C_i ← C'_i for i = 1,2,3,4
    end if
end for
```

Determining the runtime of Algorithm 6 requires the following combinatorial identity.

Lemma 5. *The following identity holds for all positive integers $n > u$:*

$$\sum_{k=u}^{n} \binom{n}{k}\binom{k}{u} = 2^{n-u}\binom{n}{u}.$$

Proof. Let $f(x) = (x+1)^n$. $f(x) = \sum_{k=0}^{n} \binom{n}{k} x^k$, by the binomial theorem. Differentiating

$f(x)$ u times yields:

$$\begin{aligned} f^{(u)}(x) &= n \cdot (n-1) \cdot \cdots \cdot (n-u+1)(x+1)^{n-u} \\ &= \sum_{k=u}^{n} \binom{n}{k} k \cdot (k-1) \cdot \cdots \cdot (k-u+1) x^{k-u}, \end{aligned}$$

which can be rewritten as:

$$\begin{aligned} f^{(u)}(x) &= \frac{n!}{(n-u)!}(x+1)^{n-u} \\ &= \sum_{k=u}^{n} \binom{n}{k} \frac{k!}{(k-u)!} x^{k-u}. \end{aligned}$$

Finally, dividing both sides by $u!$ and setting $x = 1$ yields the desired identity. □

Lemma 6 bounds the time required per iteration for Algorithm 6.

Lemma 6. *With upperbound $u \leq \frac{n}{3}$, each iteration of Algorithm 6 runs in $O^*\left(\frac{1}{2^u}\binom{n}{u}\right)$ expected time.*

Proof. The probability that $|S_1| = k$ in any given iteration is $\binom{n}{k} \cdot \frac{1}{2^n}$, as there are $\binom{n}{k}$ possible sets of k agents and the probability of S_1 being any given subset of the agents is $\frac{1}{2^n}$.

If $|S_1| = k$, then the *for* loop beginning on line 7 is executed 0 times if $k < u$ and $\binom{k}{u}$ times if $k \geq u$. Likewise, the *for* loop beginning on line 13 runs for 0 iterations if $n - k < u$ and $\binom{n-k}{u}$ iterations if $n - k \geq u$.

Summing over all possible sizes of S_1 yields an expected runtime of:

$$\begin{aligned}
& \frac{1}{2^n}\sum_{k=0}^{u-1}\binom{n}{k}\left[\binom{n-k}{u}\right]+\frac{1}{2^n}\sum_{k=u}^{n-u}\binom{n}{k}\left[\binom{k}{u}+\binom{n-k}{u}\right]+\frac{1}{2^n}\sum_{k=n-u+1}^{n}\binom{n}{k}\left[\binom{k}{u}\right] \\
= & \frac{1}{2^n}\sum_{k=0}^{n-u}\binom{n}{k}\binom{n-k}{u}+\frac{1}{2^n}\sum_{k=u}^{n}\binom{n}{k}\binom{k}{u} \\
= & \frac{2}{2^n}\sum_{k=u}^{n}\binom{n}{k}\binom{k}{u} \\
= & \frac{2}{2^n}2^{n-u}\binom{n}{u} \\
= & O^*\left(\frac{1}{2^u}\binom{n}{u}\right).
\end{aligned}$$

□

Lemma 7. *In Algorithm 6, the probability that $|S_1| \leq \frac{n}{3}$ or $|S_2| \leq \frac{n}{3}$ is $O\left(\left(\frac{19}{20}\right)^n\right)$.*

Proof. The probability that S_1 or S_2 contain no more than $\frac{n}{3}$ agents is:

$$\begin{aligned}
& \frac{1}{2^n}\sum_{k=0}^{n/3}\binom{n}{k}+\frac{1}{2^n}\sum_{k=2n/3}^{n}\binom{n}{k} \\
= & \frac{2}{2^n}\sum_{k=0}^{n/3}\binom{n}{k} \\
\leq & \frac{2}{2^n}\sum_{k=0}^{n/3}\binom{n}{n/3} \\
\leq & \frac{2n}{2^n}\binom{n}{n/3} \\
= & O\left(n\frac{3^n}{2^{5n/3}}\right)=O\left(\left(\frac{19}{20}\right)^n\right).
\end{aligned}$$

□

Since $\left(\frac{19}{20}\right)^n$ decreases to 0 exponentially in n, with probability tending towards 1, more than $\frac{n}{3}$ agents are placed in S_1 and S_2 in each iteration of Algorithm 6.

Lemma 8. *Let C_1^*, C_2^*, C_3^* and C_4^* be four disjoint coalitions from a set of n agents such that*

1. $|C_1^*| \geq |C_2^*| \geq |C_3^*| \geq |C_4^*|$,

2. $|C_1^*| + |C_4^*| \leq |C_1^*| + upperbound \leq \frac{n}{3}$, *and*

3. $|C_2^*| + |C_3^*| \leq |C_1^*| + upperbound \leq \frac{n}{3}$.

If ν *is monotonic, then after* $O(2^{|C_1^*|+|C_2^*|+|C_3^*|+|C_4^*|})$ *iterations the best solution found by Algorithm 6 with an upperbound of at least* $|C_3^*|$ *has value at least* $\nu(C_1^*) + \nu(C_2^*) + \nu(C_3^*) + \nu(C_4^*)$ *with probability at least* $\frac{1}{2}$.

Proof. If during some iteration of Algorithm 6, $C_1^*, C_3^* \subseteq S_1$, $C_2^*, C_4^* \subseteq S_2$, then since

$$\begin{aligned} upperbound &\geq |C_3^*| \geq |C_4^*| \text{ and} \\ |C_1^*| + upperbound &\leq \frac{n}{3}, \end{aligned}$$

there exists a coalition $|A| = upperbound$, such that $C_3^* \subseteq A$ and $C_1^* \subseteq S_1 \setminus A$. The *for* loop beginning on line 7 of Algorithm 6 will eventually find A and hence find two coalitions in S_1 of value at least $\nu(C_1^*) + \nu(C_3^*)$. Likewise, Algorithm 6 will find two coalitions in S_2 with value at least $\nu(C_2^*) + \nu(C_4^*)$. Therefore, Algorithm 6 will return a solution with value at least $\nu(C_1^*) + \nu(C_2^*) + \nu(C_3^*) + \nu(C_4^*)$.

By Lemmas 2 and 7, the probability that during at least one iteration of Algorithm 6, $C_1, C_3 \subseteq S_1$ and $C_2, C_4 \subseteq S_2$ and $|S_1|, |S_2| > \frac{n}{3}$ is greater than $1 - \frac{1}{e} - O^*\left(\left(\frac{19}{20}\right)^n\right) > \frac{1}{2}$, for n sufficiently large. □

Theorem 12. *A* $\frac{1}{2}$*-approximate coalition structure in* ν *can be found in* $O^*(2.09^n)$ *expected time.*

Proof. Construct $\nu_{\frac{1}{4}}$ in $O^*(2.09^n)$ time, then run Algorithm 4 to identify two disjoint coalitions that maximize the sum of their values, followed by Algorithm 5 for $3^{\frac{5n}{8}} = O(1.99^n)$ iterations. Finally, run Algorithm 6 for $O^*(2^{\frac{4n}{7}})$ iterations with an upper bound of $\frac{3n}{20}$.

This procedure guarantees a $\frac{1}{2}$-approximate solution with high probability.

Let CS^* be an optimal coalition structure in $v_{\frac{1}{4}}$ with the least number of coalitions. By Theorem 4, $|CS^*| \leq 7$.

Let $|C_1| \geq |C_2| \geq \cdots \geq |C_r|$ be the coalitions in CS^*. Note that $|C_{r-1}| + |C_r| > \frac{n}{4}$ otherwise, since $v_{\frac{1}{4}}$ is $\frac{n}{4}$-superadditive, $\{C_1, \cdots, C_{r-2}, C_{r-1} \cup C_r\}$ is an optimal coalition structure with fewer coalitions. Since $|C_{r-1}| \geq |C_r|$, it must be that $|C_{r-1}| \geq \frac{n}{8}$. Hence, for all $i < r$, $|C_i| \geq \frac{n}{8}$.

There are four cases to be considered.

1. $|CS^*| \leq 4$. Algorithm 4 returns a $\frac{1}{2}$-approximate solution in $O(2^n)$ time.

2. $|CS^*| = 5$. One of $\{C_1, C_2\}$ and $\{C_3, C_4, C_5\}$ must have total value at least $\frac{v(CS^*)}{2}$. If $v(C_1) + v(C_2) \geq \frac{v(CS^*)}{2}$, then Algorithm 4 returns a $\frac{1}{2}$-approximation solution. Note that $|C_1| + |C_2| \geq \frac{2n}{5}$ (as they are the two largest of the 5 coalitions in CS^*) and $|C_3| + |C_4| + |C_5| \leq \frac{3n}{5}$. Therefore, if $v(C_3) + v(C_4) + v(C_5) \geq \frac{v(CS^*)}{2}$, then Algorithm 5 returns a solution with value at least $v(C_3) + v(C_4) + v(C_5) \geq \frac{v(CS^*)}{2}$ with high probability in $O(3^{\frac{3n}{5}}) = O(1.94^n)$ iterations.

3. $|CS^*| = 6$. One of $\{C_1, C_5, C_6\}$ and $\{C_2, C_3, C_4\}$ must have total value at least $\frac{v(CS^*)}{2}$. If $\{C_1, C_5, C_6\}$ has total value at least $\frac{v(CS^*)}{2}$, then $|C_1| + |C_5| + |C_6| \leq \frac{5}{8}$, because $|C_2|, |C_3|, |C_4| \geq \frac{n}{8}$. Likewise, if $\{C_2, C_3, C_4\}$ has total value at least $\frac{v(CS^*)}{2}$, then $|C_2| + |C_3| + |C_4| \leq \frac{5}{8}$ because $|C_5| + |C_6| \geq \frac{n}{4}$ and $|C_1| \geq \frac{n}{8}$. Therefore, in either case, Algorithm 5 returns a solution with value at least $\frac{v(CS^*)}{2}$ in $O(3^{\frac{5n}{8}}) = O(1.99^n)$ time.

4. $|CS^*| = 7$. One of $\{C_1, C_2, C_3\}$ and $\{C_4, C_5, C_6, C_7\}$ must have total value at least $\frac{v(CS^*)}{2}$. Since $|C_4| + |C_5| + |C_6| + |C_7| \geq \frac{n}{2}$, then $|C_1| + |C_2| + |C_3| \leq \frac{n}{2}$. Therefore, if $\{C_1, C_2, C_3\}$ has value at least $\frac{v(CS^*)}{2}$, then Algorithm 5 will return a $\frac{1}{2}$-approximate solution in $O(3^{\frac{n}{2}}) = O(1.74^n)$ iterations.

   Otherwise, $\{C_4, C_5, C_6, C_7\}$ has value at least $\frac{v(CS^*)}{2}$ and contains at most $\frac{4n}{7}$ agents.

Notice that

$$|C_6| \leq \frac{1}{6}\left(n - |C_7|\right),$$

which implies that

$$\frac{n}{4} < |C_6| + |C_7| \leq \frac{1}{6}(n - |C_7|) + |C_7| = \frac{n}{6} + \frac{5}{6}|C_7|.$$

Thus, $|C_7| > \frac{n}{10}$ and $|C_6| < \frac{3n}{20}$. Now, observe that

$$\begin{aligned} |C_4| + |C_7| &\leq \left(\frac{4n}{7} - |C_5| - |C_6| - |C_7|\right) + |C_7| < \frac{4n}{7} - \frac{n}{4} < \frac{1}{3} \text{ and} \\ |C_5| + |C_6| &\leq \frac{1}{3}(n - |C_7|) \leq \frac{1}{3}\left(n - \frac{n}{10}\right) < \frac{n}{3}. \end{aligned}$$

Finally, note that, as $|C_5| + |C_6| + |C_7| > \frac{3n}{8}$, $|C_4| < \frac{1}{4} \cdot \frac{5n}{8} = \frac{5n}{32}$. Therefore $|C_4| + \frac{3n}{20} < \frac{n}{3}$ (where $\frac{3n}{20}$ is the upper bound used in Algorithm 6.

Therefore, Algorithm 6 finds a $\frac{1}{2}$-approximate solution in

$$2^{\frac{4n}{7}} \cdot \frac{1}{2^{\frac{3n}{20}}} \cdot \binom{n}{\frac{3n}{20}} = O^*(2.05^n)$$

time. Thus, the bottleneck step is the construction of $\nu_{\frac{1}{4}}$, which requires $O^*(2.09^n)$ time.

□

IV.3 Approximate Coalition Structure Generation in Monotonic Games

This section considers the coalition structure generation in monotonic games. The original algorithm was designed for arbitrary coalitional games (Service and Adams, 2011b); however, it has since been adapted for monotonic coalitional games.

First, a lower bound on the time complexity of any algorithm that guarantees to find a solution of quality strictly better than $\frac{1}{2}$ of the optimal is presented.

IV.3.1 Lower Bound

In general coalitional games, in order to provide any guarantee on the quality of the solution produced, every algorithm must, in the worst case, examine the value of every coalition at least once (Sandholm et al., 1999). This observation implies an immediate $\Omega(2^n)$ lower bound on the runtime of any deterministic coalition structure generation algorithm that provides guarantees on the quality of the solution it produces. However, this lower bound does not hold in monotonic games. For example, the coalition structure consisting of only the grand coalition obtains a $\frac{1}{n}$-approximation in any monotonic game. Theorem 13 provides a similar lower bound for computing the optimal coalition structure in monotonic games.

Theorem 13. *In a coalitional game (N, v), in order to find a coalition structure with value greater than $\frac{1}{2}$ of the optimal, any randomized or deterministic algorithm must make $\Omega\left(\frac{2^n}{\sqrt{n}}\right)$ queries to v.*

Proof. The proof employs Yao's minimax principle (Yao, 1977). Yao's minimax principle states that the expected number of queries made by any randomized algorithm is at least the lowest expected number of queries made by the best deterministic algorithm on any fixed distribution over coalitional games.

Let n be even and define a probability distribution, P, over coalitional games as follows. Let A be the set of $\Theta\left(\frac{2^n}{\sqrt{n}}\right)$ unordered pairs of coalitions $(C, N \setminus C)$ where $|C| = \frac{n}{2}$. A random coalitional game (N, v) is constructed as follows. Draw, uniformly at random, a single pair $(S, N \setminus S)$ from A. Define v_S as:

$$v_S(C) = \begin{cases} 1 & \text{if } |C| > \frac{n}{2} \\ 1 & \text{if } C = S \text{ or } C = N \setminus S \\ 0 & \text{otherwise.} \end{cases}$$

Thus, in every coalitional game in P, every coalition with fewer than $\frac{n}{2}$ agents has value 0 and every coalition with greater than $\frac{n}{2}$ agents has value 1. Also note that all but two coalitions with $\frac{n}{2}$ agents have value 0. The two coalitions with $\frac{n}{2}$ agents that have value 1 are disjoint and hence form a coalition structure with value 2. All such games are clearly monotonic. Finally, note that any two distinct games drawn from P differ only on the values of the coalitions in the optimal coalition structure in that game and that no two distinct games drawn from P have the same optimal coalition structure.

Let D be any fixed deterministic algorithm. Notice that finding a $\frac{1}{2}$-approximate solution is equivalent to finding a coalition $|C| = \frac{n}{2}$ of value 1. Hence, it is sufficient to consider the expected number of queries, over problems drawn from P, before such a coalition C is found. Since D is deterministic, the queries made by D on any problem instance from P is the same up until the point at which such a coalition C is found. Consider only those queries D makes regarding coalitions of size $\frac{n}{2}$. Given that the particular coalitions C and $N \setminus C$ are selected uniformly at random, we see that the expected number of queries made by D regarding coalitions of size $\frac{n}{2}$ is

$$\frac{\binom{n}{n/2}}{2} = \Theta\left(\frac{2^n}{\sqrt{n}}\right).$$

Hence, by Yao's minimax theorem, the expected number of queries made by any randomized algorithm that guarantees solution with value greater than $\frac{1}{2}$ of the optimal is $\Omega\left(\frac{2^n}{\sqrt{n}}\right)$. □

IV.3.2 Approximate Coalition Structure Generation

An algorithm for approximate coalition structure generation in monotonic games is now presented. The presented algorithm is based on the following observation formalized in Lemma 9. Fix any instance of the coalition structure generation problem and any integer $r \geq 2$. There is a subproblem S consisting of $\lfloor \frac{n}{r} \rfloor$ or fewer agents, such that the value of

the optimal solution over S, plus the value of the highest valued coalition over the agents in $N \setminus S$ is at least $\frac{1}{r}$ times the value of the optimal solution to the original problem. In monotonic coalitional games, the highest valued coalition over the agents in $N \setminus S$ is simply the coalition $C = N \setminus S$.

Recall the following notation introduced in Chapter 2:

1. Let $opt(C)$ denote the maximum value over all coalition structures that can be formed from the agents in C.

2. Let $sol(C)$ denote the optimal coalition structure for the subproblem consisting of only those agents in C.

The presented algorithm guarantees an $\frac{1}{r}$-approximation by generating the optimal coalition structure, via dynamic programming, for all subproblems consisting of $\lfloor \frac{n}{r} \rfloor$ or fewer agents. That is, $opt(C)$ and $sol(C)$ are computed for all $|C| \leq \lfloor \frac{n}{r} \rfloor$. The algorithm returns the coalition structure $sol(C) \cup \{N \setminus C\}$ for the set of agents C, $|C| \leq \lfloor \frac{n}{r} \rfloor$, which maximizes $opt(C) + v(N \setminus C)$. It is shown that for all $r \geq 2$, this approach results in a $\frac{1}{r}$-approximate solution.

Table IV.3 provides the runtime of the presented algorithm for a number of constant factor performance guarantees. In addition to being able to guarantee various constant factor approximation guarantees, the presented algorithm can obtain non-constant factor approximation guarantees in substantially less time. For example, a $\frac{1}{\sqrt{n}}$-approximate solution can be obtained in $O((1+\varepsilon)^n 2^{\sqrt{n}\log n})$ time, for all $\varepsilon > 0$.

Table IV.3: Example approximation ratios achievable by the presented algorithm and the corresponding runtimes running times. The first row provides the approximation ratio in terms of fraction of the optimal and the second row provides the corresponding runtime.

Approximation Ratio	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{7}$	$\frac{1}{8}$
Run time	$O^*(1.895^n)$	$O^*(1.762^n)$	$O^*(1.664^n)$	$O^*(1.59^n)$

A $\frac{1}{r}$-approximation of the optimal coalition structure is obtained by running the standard dynamic programming algorithm that generates partial solutions only for those coali-

tions of $\frac{n}{r}$ or fewer agents. As the search progresses, we track the subset of agents C, that maximizes the sum of the value of the optimal coalition structure over C, plus the value of $N \setminus C$. The pseudocode is provided in Algorithm 7.

Algorithm 7 Approximation Algorithm

```
1:  Best = ∅
2:  for k = 1 to n/r do
3:     for C ⊆ N, |C| = k do
4:        opt(C) ← v_C
5:        for C' ⊂ C, |C'| ≤ (1/2)|C| do
6:           if opt(C') + opt(C \ C') > opt(C) then
7:              opt(C) ← opt(C') + opt(C \ C')
8:           end if
9:        end for
10:       if opt(C) + v(N \ C) > opt(Best) + v(N \ Best) then
11:          Best ← C
12:       end if
13:    end for
14: end for
```

Best in Algorithm 7 tracks the subset of agents C that maximizes the sum $opt(C) + v(N \setminus C)$ and is initialized to the empty set.

The proof of the factor r performance ratio obtained by this algorithm is based on partitioning the coalitions in any optimal solution OPT in such a way as to minimize the imbalance of the contributions of each partition to the overall value of OPT. Formally, let OPT be any optimal coalition structure to a fixed problem instance. Let $P = \{P_1, \cdots, P_r\}$ be a partitioning of OPT into r parts (i.e., each P_i is a set of coalitions and each P_j, P_k, $k \neq j$ are disjoint). The P_i's are allowed to be empty. Let $V(P_i)$ be the sum of the coalition values of the coalitions in P_i. Thus, $V(OPT) = V(P_1) + \cdots + V(P_r)$. Define the imbalance of P, $IB(P)$, to be:

$$IB(P) = \sum_{P_i \in P} \left(V(P_i) - \frac{V(OPT)}{r} \right)^2.$$

In a perfectly balanced partitioning of OPT, each partition contributes exactly $\frac{V(OPT)}{r}$ to the overall value of OPT. Assume, without loss of generality, that $V(P_1) \geq V(P_2) \geq \cdots \geq$

$V(P_r)$. Thus, P_1 is not empty and $V(P_1) \geq \frac{V(OPT)}{r}$.

Lemma 9. *Let $P = \{P_1, \cdots, P_r\}$ be a partitioning of OPT with the minimum amount of imbalance. Let C be a coalition in P_i of maximum value, then $V(P_k \cup \{C\}) \geq V(P_i)$, for all $k \geq i$.*

Proof. The lemma is trivially true for $i = r$. Assume that $i < r$.

The lemma is proven for $k = r$, since $V(P_j) \geq V(P_r)$ for $j < r$, the general case will follow for all $k \geq i$.

If $v(C) = 0$, then $V(P_i) = 0$ and the lemma is trivially true. Thus, assume that $v(C) > 0$.

Let $P' = \{P_1, \cdots, P_i \setminus \{C\}, \cdots, P_r \cup \{C\}\}$ be the partitioning of OPT formed by removing coalition C from P_i and adding it to P_r. Since P has the minimum imbalance over all partitions of OPT, $IB(P') \geq IB(P)$. Since all but the ith and last terms of $IB(P')$ and $IB(P)$ are equal, the difference of $IB(P')$ and $IB(P)$ can be written as:

$$\begin{aligned}
IB(P') - IB(P) &= \left(V(P_i \setminus \{C\}) - \frac{V(OPT)}{r}\right)^2 + \left(V(P_r \cup \{C\}) - \frac{V(OPT)}{r}\right)^2 \\
&\quad - \left(V(P_i) - \frac{V(OPT)}{r}\right)^2 - \left(V(P_r) - \frac{V(OPT)}{r}\right)^2 \\
&= \left((V(P_i) - V(\{C\})) - \frac{V(OPT)}{r}\right)^2 + \left((V(P_r) + V(\{C\})) - \frac{V(OPT)}{r}\right)^2 \\
&\quad - \left(V(P_i) - \frac{V(OPT)}{r}\right)^2 - \left(V(P_r) - \frac{V(OPT)}{r}\right)^2 \\
&= (V(P_i) - V(\{C\}))^2 - 2(V(P_i) - V(\{C\}))\left(\frac{V(OPT)}{r}\right) \\
&\quad + (V(P_r) + V(\{C\}))^2 - 2(V(P_r) + V(\{C\}))\left(\frac{V(OPT)}{r}\right) \\
&\quad - V(P_i)^2 + 2V(P_i)\left(\frac{V(OPT)}{r}\right) - V(P_r)^2 + 2V(P_r)\left(\frac{V(OPT)}{r}\right) \\
&= V(C)^2 - 2V(P_i)V(C) + V(C)^2 + 2V(P_r)V(C) \\
&= 2V(C)\left((V(P_r) + V(C)) - V(P_i)\right) \\
&= 2V(C)\left(V(P_r \cup \{C\}) - V(P_i)\right).
\end{aligned}$$

Since $IB(P') - IB(P) \geq 0$ and $V(C) > 0$, it follows that $V(P_r \cup \{C\}) \geq V(P_i)$. □

The factor r performance ratio obtained by Algorithm 7 can now be proven.

Theorem 14. *After Algorithm 7 terminates, the value of the coalition structure CS is within a factor $\frac{1}{r}$ of the optimal.*

Proof. Let $P = \{P_1, \cdots, P_r\}$ be a partitioning of OPT of minimum imbalance. Again assume that $V(P_1) \geq V(P_2) \geq \cdots \geq V(P_r)$.

By the pigeonhole principle, since there are n agents and only r partitions, at least one of the partitions, P_i consists of at most $\lfloor \frac{n}{r} \rfloor$ agents.

If P_1 has at most $\lfloor \frac{n}{r} \rfloor$ agents, then Algorithm 7 will generate a coalition structure on the agents in P_1 that has a value at least $V(P_1) \geq \frac{V(OPT)}{r}$, as desired.

Let $C = \cup_{S \in P_i} S$. Since Algorithm 7 determines the optimal coalition structure for all subproblems consisting of $\lfloor \frac{n}{r} \rfloor$ or fewer agents, $opt(C)$ will be computed. Further, $opt(C) = V(P_i)$, otherwise OPT is not an optimal coalition structure. Let S be the highest valued coalition in P_1. Since v is monotonic, $v(N \setminus C) \geq v(S)$. By Lemma 9:

$$opt(C) + v(N \setminus C) \geq opt(C) + v(S) = V(P_i) + v(S) \geq V(P_1) \geq \frac{V(OPT)}{r}.$$

However, $opt(Best) + v(N \setminus Best) \geq opt(C) + v(N \setminus C)$. Thus, Algorithm 7 returns a $\frac{1}{r}$-approximate solution. □

The runtime of Algorithm 7 is given in Theorem 15.

Theorem 15. *Algorithm 7 runs in $O^*\left(\left(\frac{r(2(r-1))^{\frac{1}{r}}}{r-1}\right)^n\right)$ time.*

Proof. Follows directly from Theorem 7 with $k = 1$. □

Thus, an $\frac{1}{r}$-approximate solution to the coalition structure generation problem in monotonic coalitional games can be obtained in $O^*\left(\left(\frac{r(2(r-1))^{\frac{1}{r}}}{r-1}\right)^n\right)$.

IV.4 Discussion

This chapter presented several approximation algorithms for the coalition structure generation problem in both arbitrary coalitional games as well as monotonic coalitional games. This chapter also presented the first randomized coalition structure generation algorithms. The randomized approximation algorithms have better theoretical guarantees than their deterministic counterparts.

The presented approximation algorithm for monotonic games is capable of generating coalition structures with a value within a constant factor of the optimal in less time than $O^*(2^n)$ time. That is, constant factor approximations can be obtained in monotonic games without examining the value of each coalition. To the best of the author's knowledge this is the first algorithm designed specifically for monotonic coalitional games. A lower bound of $\Omega\left(\frac{2^n}{\sqrt{n}}\right)$ on computing a $\frac{1}{2}$-approximate coalition structure in monotonic games is also provided.

CHAPTER V

Empirical Comparison

This chapter presents an empirical comparison of the approximation algorithms presented in Chapter IV. The empirical comparison is divided into three components. First, the performance of the deterministic and randomized approximation algorithms are compared over four different problem distributions. Second, the performance of the deterministic approximation algorithms are compared to the performance of existing techniques in the literature. Finally, an empirical study of the performance of the monotonic approximation algorithm is presented. First, the problem distributions employed throughout this chapter are defined in Section V.1.

All of the empirical results that report runtimes were performed on a Dell Precision T3500 with an Intel Xeon CPU running at 3.33GHz with 6 GB of RAM.

V.1 Problem Distributions

All of the empirical results presented in this chapter are with respect to four problem distributions from the literature and two newly introduced distributions (the Monotonic and SVA distributions). This section describes each problem distribution.

1. Normal Distribution: the value of each coalition C was set to $|C| \cdot N(1.0, 0.01)$, where $N(1.0, 0.01)$ represents a normal distribution with mean 1 and variance 0.01.

2. Uniform Distribution: each coalition C is assigned a value drawn uniformly between 0 and $|C|$.

3. Modified Uniform Distribution: each coalition C is assigned a value drawn uniformly between 0 and $10 \cdot |C|$; however, each coalition's value is increased by a random number drawn uniformly between 0 and 50 with 20% probability.

4. Normally Distributed Coalition Structures (NDCS) (Rahwan et al., 2009b): the value of each is coalition C is drawn from a normal distribution with mean $|C|$ and standard deviation $\sqrt{|C|}$.

5. Monotonic Distribution: the value of each coalition, C, was first assigned as in the Uniform distribution, then the value of C was set to $\max_{S \subseteq C} v(S)$ in order to make v monotonic.

6. Single Valuable Agent (SVA)[1]: the value of a coalition C is defined as follows. If $N = \{1, \cdots, n\}$ (i.e., the agents are identified with the first n positive integers) then:

$$v(C) = \begin{cases} 200 & \text{if } i \in C \\ 0 & \text{otherwise.} \end{cases}$$

The runtimes of the presented deterministic and randomized approximation algorithms, Sandholm et al.'s algorithm, and Dang and Jennings' algorithm do not depend on the particular problem distribution. However, the IP algorithm prunes subspaces based upon the particular values of individual coalitions. Hence, the runtime of the IP algorithm will differ from one problem distribution to the next.

V.2 Approximate Coalition Structure Generation

This section compares the approximation performance of the algorithms presented in this dissertation as well as others existing in the literature.

V.2.1 Randomized vs. Deterministic

This subsection presents a comparison between the deterministic and randomized approximation algorithms. The performance of both the deterministic and randomized algorithms

[1]The value 200 in the definition of the SVA problem distribution can be replaced by any other positive integer without changing the theoretical or empirical results.

are compared over the Uniform, Normal, NDCS, and Modified Uniform problem distributions.

Solution Quality

Table V.1: The average utilities generated by the randomized and deterministic algorithms for five different approximation ratios on three problem distributions consisting of 25 agents.

	Approximation Ratio	2/3	3/5	1/2	2/5	1/3
Uniform	Randomized	24.99	24.99	24.99	N/A	24.99
	Deterministic	24.99	N/A	24.99	24.99	24.99
Normal	Randomized	33.9	33.84	33.81	N/A	33.81
	Deterministic	33.98	N/A	33.91	33.86	33.86
Modified Uniform	Randomized	669.43	580.33	523.42	N/A	429.05
	Deterministic	654.44	N/A	573.24	477.751	448.24
NDCS	Randomized	66.82	61.38	59.06	N/A	58.58
	Deterministic	66.05	N/A	58.3	55.38	55.38

Table V.1 provides the average utility of the solutions generated by both the deterministic and the randomized algorithms. Recall that no deterministic $\frac{3}{5}$-approximation algorithm exists and no randomized $\frac{2}{5}$-approximation algorithm exists. All data points are averaged over 50 independent trials. In general, the average utility of solutions generated by the randomized algorithm was comparable to that of the deterministic algorithm.

It is interesting to note that during some of the runs the randomized algorithm performed better than the deterministic algorithm. For example, the average performance of the randomized $\frac{1}{3}$-approximation algorithm was slightly higher then the average performance of the deterministic $\frac{1}{3}$-approximation for the Modified Uniform problem distribution. This different in performance is to be expected though. Consider the $\frac{2}{3}$-approximation algorithms. Assume that the optimal coalition structure consists of three equally sized, and equally valued, coalitions, C_1, C_2, and C_3. Further assume that all other coalitions have value 0. The deterministic $\frac{2}{3}$-approximation algorithm will simply return an approximate coalition structure consisting of two of these coalitions, say C_1 and C_2. However, it is possible that

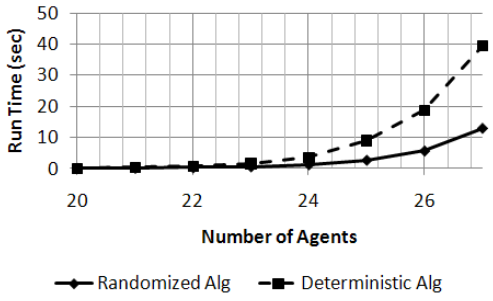
the randomized $\frac{2}{3}$-approximation algorithm will return a coalition structure consisting of C_1, C_2 and C_3, thus returning a solution with value strictly greater than that returned by the deterministic algorithm. While this example is artificial, it illustrates one type of situation where the randomized algorithms can outperform their deterministic counterparts (i.e., situations where the randomized algorithm has some probability of returning an optimal solution, while the deterministic algorithm is unable too).

Runtime

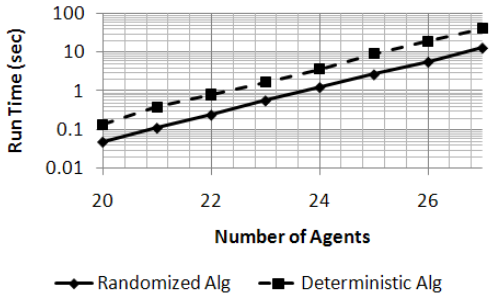
This subsection compares the runtime of the deterministic and randomized algorithm over a number of problem sizes ranging from 20 to 27 agents. Each data point is averaged over 50 independent trials. Separate graphs are presented for each approximation ratio. In addition, graphs are presented on both a logarithmic and non-logarithmic scale. As the runtime of both the randomized and deterministic algorithms does not depend on the problem instance (i.e., the particular coalition utilities) the reported runtimes are only for the Modified Uniform Distribution.

Graphs of the runtime of the deterministic and randomized $\frac{1}{3}$-approximation algorithms are presented in Figure V.1. For 27 agents, the randomized $\frac{1}{3}$-approximation algorithm required less than 33% of the time used by the deterministic $\frac{1}{3}$-approximation algorithm.

Figure V.1: Comparison of the runtimes of the deterministic and randomized $\frac{1}{3}$-approximation algorithms on a variety of problem sizes.



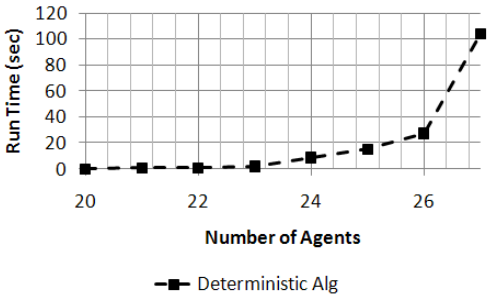
(a) Runtime in seconds of the both the deterministic and randomized $\frac{1}{3}$-approximation algorithms.



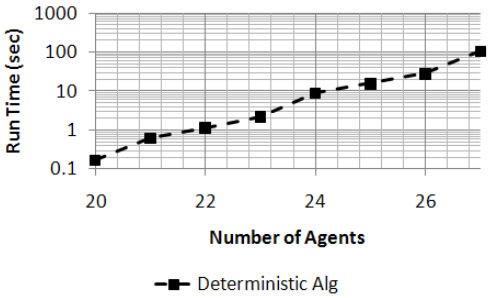
(b) Runtime in seconds of the both the deterministic and randomized $\frac{1}{3}$-approximation algorithms on a logarithmic scale.

A graph of the runtime of the deterministic $\frac{2}{5}$-approximation algorithm is presented in Figure V.2. Recall that no randomized $\frac{2}{5}$-approximation algorithm exists currently.

Figure V.2: Graph of the runtime of the deterministic $\frac{2}{5}$-approximation algorithm on a variety of problem sizes.



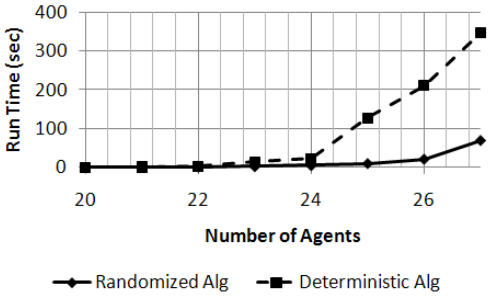
(a) Runtime in seconds of the deterministic $\frac{2}{5}$-approximation algorithm.



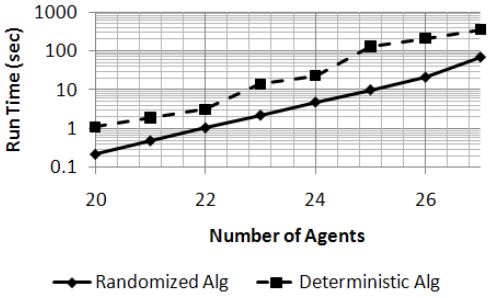
(b) Runtime in seconds of the deterministic $\frac{2}{5}$-approximation algorithm on a logarithmic scale.

Graphs of the runtime of the deterministic and randomized $\frac{1}{2}$-approximation algorithms are presented in Figure V.3. For 27 agents, the randomized $\frac{1}{2}$-approximation algorithm required less than 20% of the time that the deterministic $\frac{1}{2}$-approximation algorithm required.

Figure V.3: Comparison of the runtimes of the deterministic and randomized $\frac{1}{2}$-approximation algorithms on a variety of problem sizes.



(a) Runtime in seconds of the both the deterministic and randomized $\frac{1}{2}$-approximation algorithms.

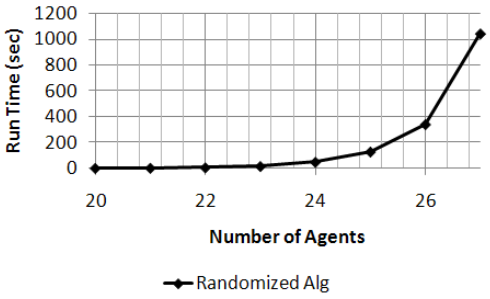


(b) Runtime in seconds of the both the deterministic and randomized $\frac{1}{2}$-approximation algorithms on a logarithmic scale.

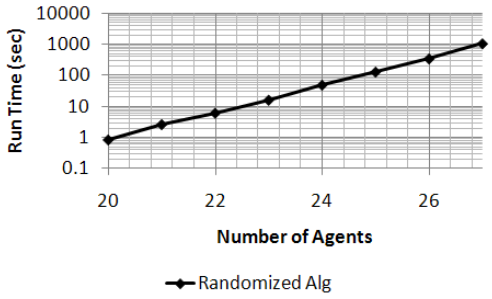
A graph of the runtime of the randomized $\frac{3}{5}$-approximation algorithm is presented in Figure V.4. Recall that there is currently no deterministic $\frac{3}{5}$-approximation algorithm.

Graphs of the runtime of the deterministic and randomized $\frac{2}{3}$-approximation algorithms

Figure V.4: Graph of the runtime of the randomized $\frac{3}{5}$-approximation algorithm on a variety of problem sizes.



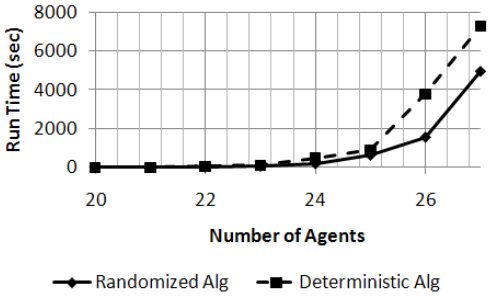
(a) Runtime in seconds of the randomized $\frac{3}{5}$-approximation algorithm.



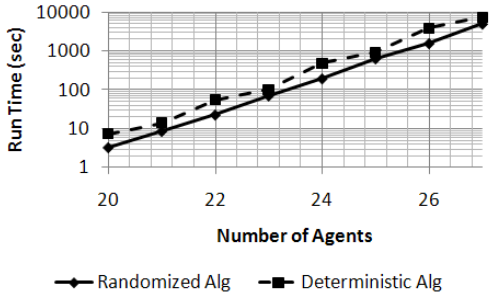
(b) Runtime in seconds of the randomized $\frac{3}{5}$-approximation algorithm on a logarithmic scale.

are presented in Figure V.5. For 27 agents, the randomized $\frac{2}{3}$-approximation algorithm ran in less than 67% of the time required by the deterministic $\frac{2}{3}$-approximation algorithm.

Figure V.5: Comparison of the runtimes of the deterministic and randomized $\frac{2}{3}$-approximation algorithms on a variety of problem sizes.



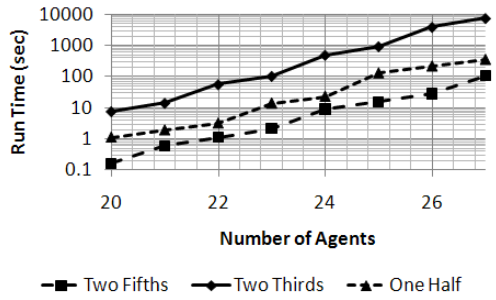
(a) Runtime in seconds of the both the deterministic and randomized $\frac{2}{3}$-approximation algorithms.



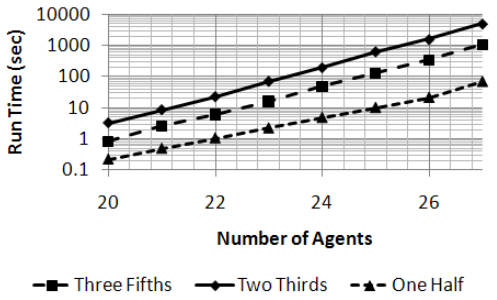
(b) Runtime in seconds of the both the deterministic and randomized $\frac{2}{3}$-approximation algorithms on a logarithmic scale.

Subfigure V.7(a) presents the runtime of the deterministic approximation algorithm over the $\frac{2}{3}$-,$\frac{2}{5}$-, and $\frac{1}{2}$-approximation guarantees, for problems consisting of between 20 and 27 agents. Subfigure V.7(b) presents the a similar comparison of the randomized approximation algorithm for $\frac{2}{3}$-,$\frac{3}{5}$-, and $\frac{1}{2}$-approximation guarantees. Both graphs in Figure V.6 are on a logarithmic scale. As expected, the lower the approximation guarantee, the faster the deterministic and randomized algorithms run.

Figure V.6: Comparison different approximation ratios for both the deterministic and the randomized approximation algorithms.



(a) Comparison of the various approximation ratios for the deterministic approximation algorithm.



(b) Comparison of the various approximation ratios for the randomized approximation algorithm.

V.2.2 Deterministic Approximation Algorithm versus Existing Algorithms

This subsection compares the performance of the presented deterministic approximation algorithms to existing algorithms from the literature. In particular, the following algorithms are compared:

1. The deterministic approximation algorithm for arbitrary coalitional games presented in Section IV.1 of Chapter IV.

2. The Integer Partition (IP) algorithm (Rahwan et al., 2009b).

3. The Improved Dynamic Programming - Integer Partition (IDP-IP) algorithm (Rahwan and Jennings, 2008a).

4. Sandholm et al.'s algorithm (Sandholm et al., 1999), referred to as Sandholm's algorithm.

5. Dang and Jenning's algorithm (Dang and Jennings, 2004).

Talal Rahwan provided the implementation of the IP and IDP-IP algorithms employed throughout all empirical studies presented in this chapter. The implementation of the IP and IDP-IP algorithms did not provide the time taken by the IP/IDP-IP algorithm to guarantee

a given quality bound. However, the implementation did provide the number of expansions performed until a given quality bound was achieved. The results in this chapter, approximate the time required by the IP/IDP-IP algorithm to achieve a given solution quality guarantee based on the ratio of the number of expansions performed until the quality bound was achieved to the total number of expansions. This value is multiplied by the total time required by the algorithm. That is, if 100 expansions were made prior to achieving a bound of $\frac{1}{2}$, a total of 1000 expansions were made before the algorithm terminated, and the algorithm ran in 10 seconds, then the estimated time required before achieving a bound of $\frac{1}{2}$ is: $\frac{100}{1000} \cdot 10$, or 1 second.

The empirical results are presented in two parts. First, the performance of each algorithm is compared on standard problem distributions from the literature. Under these distributions we find that the IP algorithm performs quite well in comparison to existing algorithms and in comparison to the approximation algorithm presented in this dissertation.

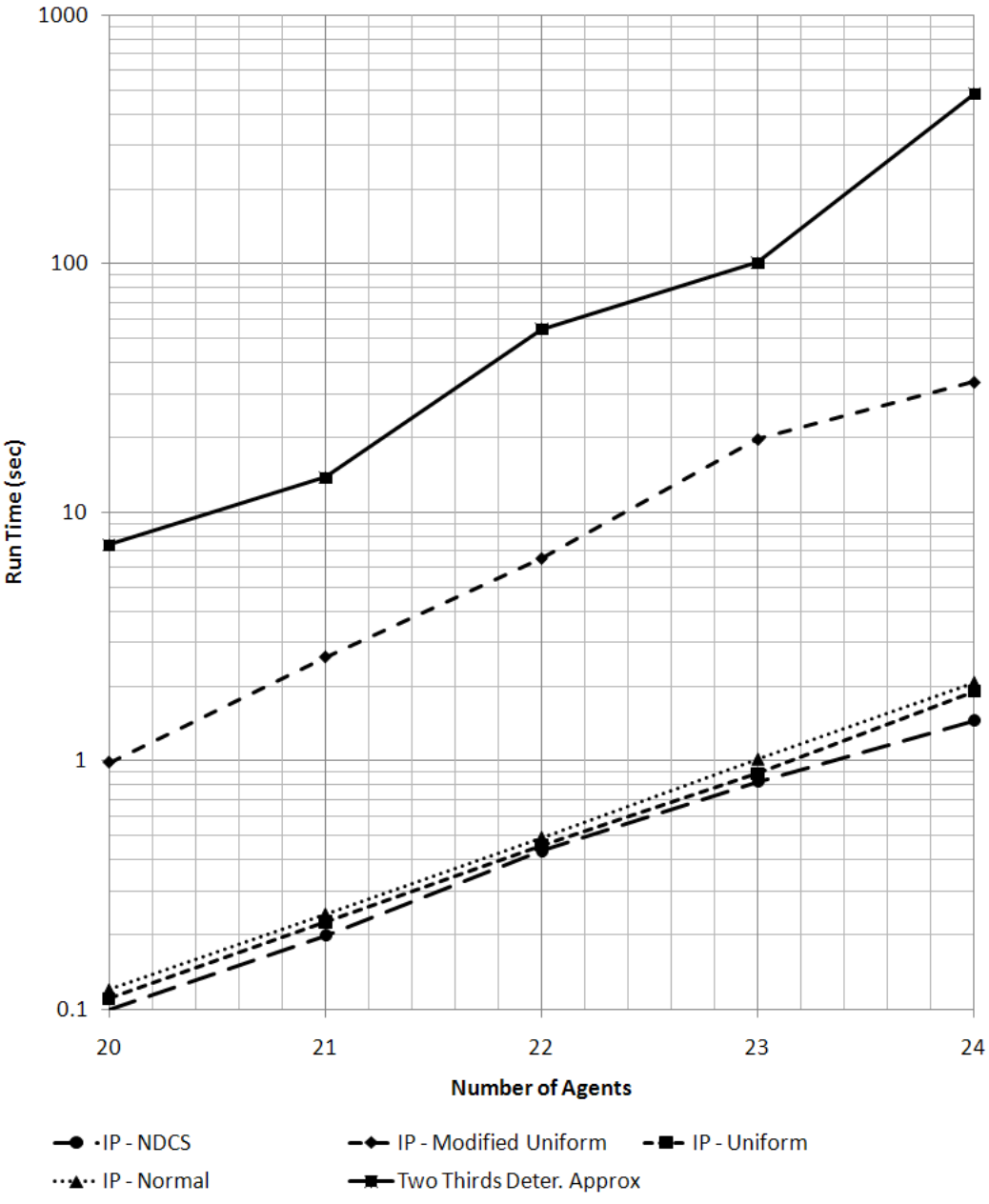
Standard Problem Distributions

We begin be comparing the performance of the presented deterministic approximation algorithm to the performance of the existing algorithms on standard problem distributions from the literature, defined in Section V.1. We delay comparing the IP algorithm to Sandholm's and Dang and Jenning's algorithms until we consider the Single Valuable Agent problem distribution.

A comparison of the time required to find a $\frac{2}{3}$-approximate solution by both the IP algorithm and the deterministic approximation algorithm the Uniform, Normal, NDCS, and Modified Uniform problem distributions is presented in Figure V.7 . The IP algorithm generates a $\frac{2}{3}$-approximate solution for all four problem distributions in much less time than the presented algorithm due to its ability to prune large portions of the search space. The IP algorithm generated a $\frac{2}{3}$-approximate solution immediately after scanning the input on many problem runs on the NDCS, Uniform, and Normal problem distributions. The

results presented in Figures V.7 and V.8 are averaged over 50 independent trials. As the deterministic approximation algorithm's runtime does not depend on particular problem instance only a single plot is shown.

Figure V.7: Comparison of the time required to find a $\frac{2}{3}$-approximate solution by the IP algorithm and the deterministic approximation algorithm on a number of different problem distributions.



A comparison of the time required by the IP algorithm to find an optimal solution for

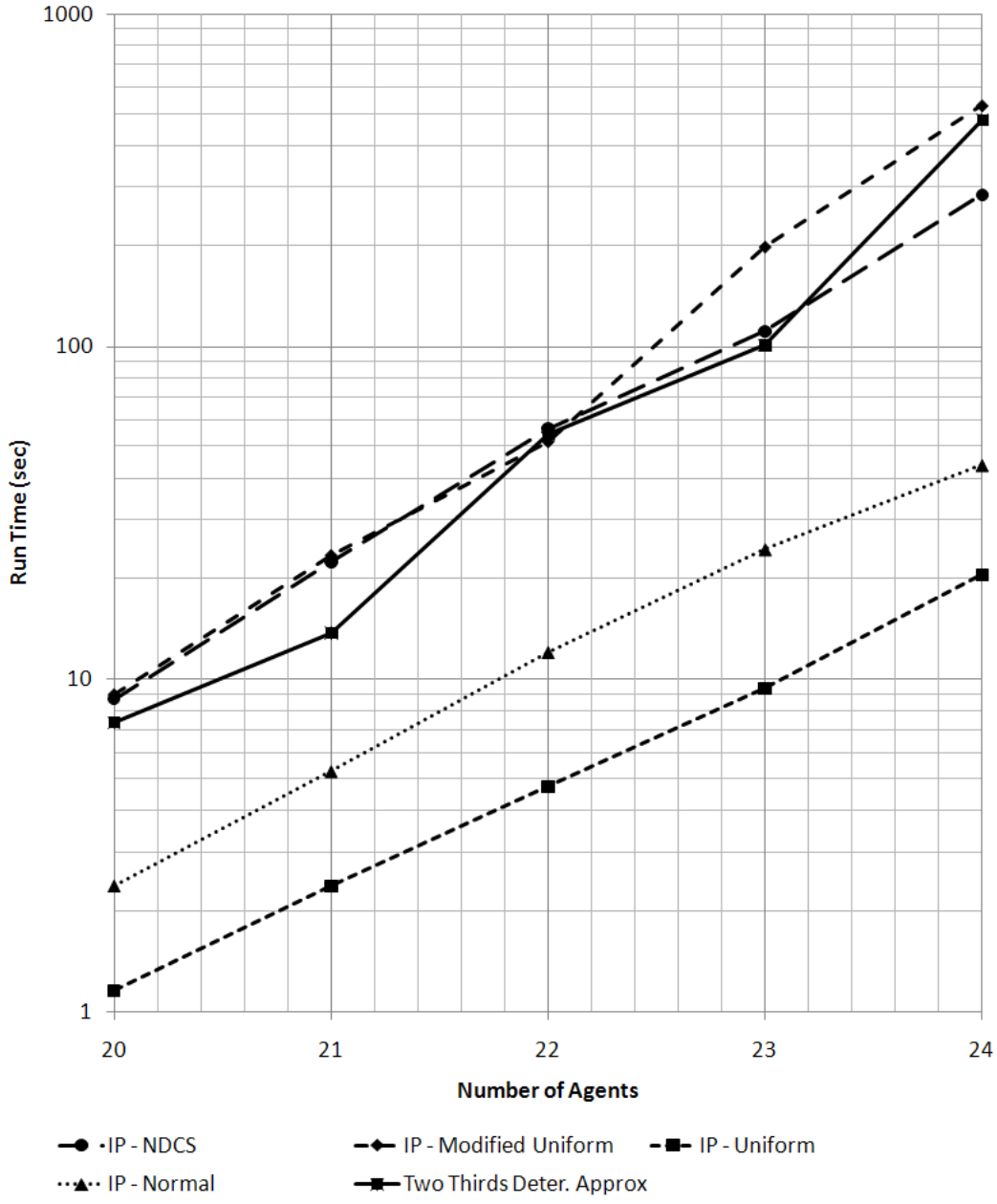
each problem distribution compared to the time required by the $\frac{2}{3}$-approximation algorithm is presented in Figure V.8. We find that for each problem distribution, the IP algorithm is able to find an optimal solution in roughly the same amount of time, or significantly faster in the case of the Uniform and Normal distributions, than required by the $\frac{2}{3}$-approximation algorithm.

The average value of the coalition structure generated by both the IP algorithm and the deterministic approximation algorithm on all four problem distributions is provided in Table V.2. The average value presented for the IP algorithm is averaged over the values of the first coalition structure found that is guaranteed to be a $\frac{2}{3}$-approximate solution. The IP algorithm and the deterministic approximation algorithm performed roughly the same in terms of solution quality on the Uniform and Normal distributions. The largest differences in solution quality occurred in the Modified Uniform and NDCS distributions. On problems from the Modified Uniform and NDCS distributions, the $\frac{2}{3}$-approximation algorithm produced higher quality solutions. However, the IP algorithm guaranteed a $\frac{2}{3}$-approximate solution much quicker. The utilities in Table V.2 are averaged over the utility of the first $\frac{2}{3}$-approximate solution guaranteed by the IP algorithm. The IP algorithm improved upon this solution with additional time. The data points in Table V.2 are averaged over 50 independent trials.

A comparison of the runtimes of the deterministic $\frac{2}{3}$-approximation algorithm, Sandholm's algorithm, and Dang and Jenning's algorithm on problems ranging from 13 to 17 agents is presented in Figure V.9. The deterministic approximation algorithm generates a $\frac{2}{3}$-approximation in significantly less time than required by Sandholm's algorithm to guarantee a $\frac{1}{2}$-or $\frac{1}{3}$-approximate solution. Similarly, the deterministic $\frac{2}{3}$-approximation algorithms requires less time than is required by Dang and Jenning's algorithm to guarantee a $\frac{1}{3}$-approximate solution. As the time required by each algorithm depends only on the number of agents and not on the particular problem distribution, only one plot is provided.

The average value of the coalition structure generated by Sandholm's algorithm, Dang

Figure V.8: Comparison of the time required to find an optimal solution by the IP algorithm and the time required by the deterministic $\frac{2}{3}$-approximation algorithm on a number of different problem distributions.

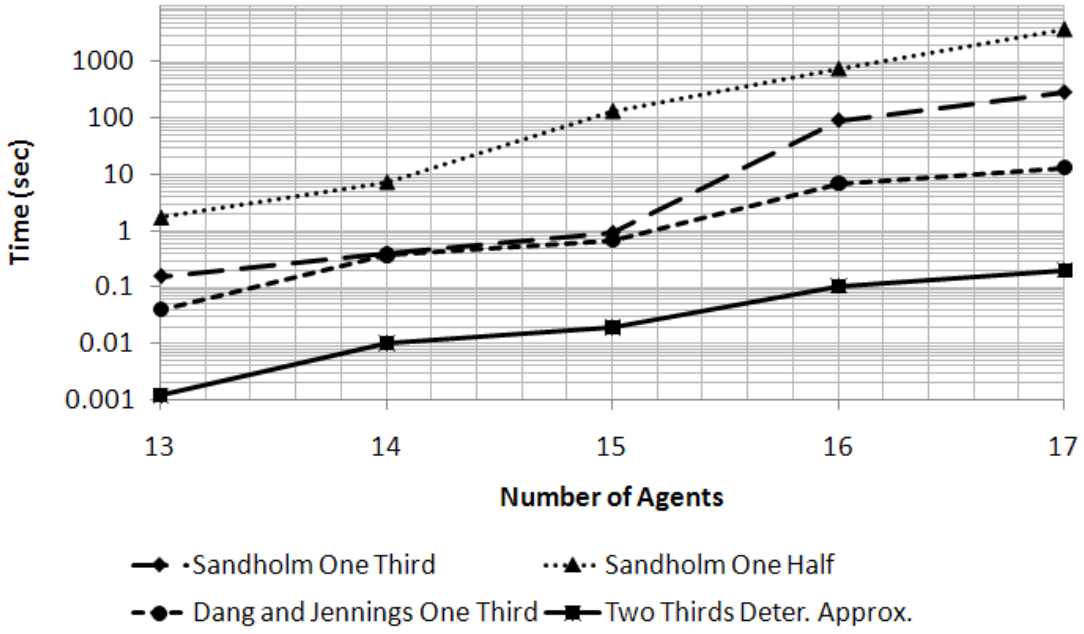


and Jenning's algorithm, and the deterministic approximation algorithm on all four problem distributions is provided in Table V.3. The average values presented for Sandholm's algorithm and Dang and Jenning's algorithm are averaged over the values of the first coalition structure found that is guaranteed to be a $\frac{1}{2}$-approximate solution. The deterministic

Table V.2: The average utilities generated by the IP algorithm and the deterministic $\frac{2}{3}$-approximation algorithm on the Uniform, Normal, Modified Uniform, and NDCS problem distributions consisting of 24 agents. The utility for the IP algorithm is averaged over the utility of the current best known coalition structure at the moment when the IP algorithm guarantees a $\frac{2}{3}$-approximate solution.

Problem Distribution	Algorithm	Solution Value
Uniform	IP Algorithm	23.9933
	Approx. Algorithm	23.9944
Normal	IP Algorithm	32.4052
	Approx. Algorithm	32.5946
Modified Uniform	IP Algorithm	535.909
	Approx. Algorithm	635.616
NDCS	IP Algorithm	49.5294
	Approx. Algorithm	64.4605

Figure V.9: Comparison of the time required by the deterministic $\frac{2}{3}$-approximation algorithm, Sandholm's Algorithm to find $\frac{1}{3}$- and $\frac{1}{2}$-approximation solution, and Dang and Jenning's algorithm to find $\frac{1}{3}$-approximate solution.



$\frac{1}{2}$-approximation algorithm, Sandholm's algorithm, and Dang and Jenning's algorithm performed comparably in terms of solution quality on the Uniform and Normal distributions.

However, on Modified Uniform and NDCS problem distributions the solution quality varied. Sandholm's and Dang and Jenning's algorithms both produced higher quality solutions than the $\frac{1}{2}$-approximation algorithm. Sandholm's and Dang and Jenning's algorithms performed similarly on the NDCS problem distribution; however, Sandholm's algorithm performed slightly better than Dang and Jenning's on the Modified Uniform problem distribution.

The data points in Figure V.9 and Table V.3 for Sandholm and Dang and Jenning's algorithm are averaged over 25 independent trials and the data points for the deterministic approximation algorithm are averaged over 50 trials.

Table V.3: The average utilities generated by Sandholm's algorithm, Dang and Jenning's algorithm and the deterministic approximation algorithm for $\frac{1}{3}$- and $\frac{1}{2}$-approximation ratios on the problem distributions for 17 agents. The utility for Sandholm's and Dang and Jenning's algorithms is averaged over the utility of the current best known coalition structure at the moment when each algorithm guarantees the stated approximation ratio.

Problem Distribution	Algorithm	$\frac{1}{2}$	$\frac{1}{3}$
Uniform	Sandholm's Algorithm	16.9544	16.9544
	Dang and Jenning's Algorithm	N/A	16.9672
	Approx. Algorithm	16.9594	16.9573
Normal	Sandholm's Algorithm	22.1002	22.1002
	Dang and Jenning's Algorithm	N/A	22.4751
	Approx. Algorithm	22.3946	22.2432
Modified Uniform	Sandholm's Algorithm	406.691	373.234
	Dang and Jenning's Algorithm	N/A	350.923
	Approx. Algorithm	337.572	289.175
NDCS	Sandholm's Algorithm	41.9716	39.0501
	Dang and Jenning's Algorithm	N/A	40.2455
	Approx. Algorithm	37.7656	35.7363

Single Valuable Agent

The previous results showed how the IP algorithm is often capable of generating high quality solutions remarkable quickly. However, we show that on problems from the SVA distribution, the IP algorithm is unable to generate good upper bound estimates on the value

of solutions within each subspace.

Two theoretical results regarding the performance of the IP algorithm for the SVA distribution are presented prior to the empirical results.

Theorem 16. *Consider a problem constructed from the SVA distribution. Let* $I = [i_1, \cdots, i_k]$ *be a subspace (i.e.* $i_1, \cdots, i_k$ *is an integer partition of n), the upper bound on the value of the optimal coalition structure within I derived by the IP algorithm is* $200 \cdot k$.

Proof. The upper bound derived for the coalition structures in I is $\sum_{m=1}^{k} Max(i_m)$, where $Max(i_m)$ is the maximum value of coalitions of size i_m. Since every coalition containing agent 1 has value at least 200 and for every $l \in \{1, \cdots, n\}$ there is a coalition of size l containing agent 1, $Max(l) = 200$ for each $l = 1, \cdots, n$. The bound immediately follows. □

Notice that the value of any coalition structure in a problem from the SVA distribution is exactly 200, as the agent 1 appears in exactly one coalition in any coalition structure. Even though every coalition structure has the same value, in order to be guaranteed to have identified an optimal coalition structure, the IP algorithm must search every subspace, as every subspace has an upper bound greater than the value of an optimal solution.

Corollary 17. *On a problem from the SVA distribution, to guarantee a* $\frac{1}{r}$*-approximation solution for* $r \in \mathbb{N}$*, the IP algorithm must search all subspaces that contain coalition structures consisting of greater than r coalitions.*

Proof. Assume that the IP algorithm does not search a subspace I that contains coalition structures consisting of $m > r$ coalitions, but still guarantees a $\frac{1}{r}$-approximate solution. Then the upper bound of the coalition structures within I is $200m$. Therefore, the best bound on the quality of the current solution is no greater than $\frac{1}{m}$. However, $\frac{1}{m} < \frac{1}{r}$, contradicting the fact that the IP algorithm guaranteed a $\frac{1}{r}$-approximate solution. □

The IDP-IP algorithm employs dynamic programming as a preprocessing phase and then runs the IP algorithm. During the preprocessing phase, the optimal solution to all

subproblems consisting of at most m agents are computed using the standard dynamic programming algorithm. Then IP algorithm is run using the results of the dynamic programming preprocessing phase.

Notice that, in the SVA distribution, the value optimal solution to a subproblem $S \subset N$ is 0 if agent 1 is not in S and is 200 if agent 1 is present in S. Thus, the bounds generated by the IP algorithm after the dynamic programming preprocessing phase are the same as the bounds the IP algorithm will generate without the preprocessing phase. Therefore, Corollary 17 applies to the IDP-IP algorithm as well.

The runtime data points for the IP and IDP-IP algorithm on problems from the SVA distribution are averaged over 10 runs. Since for each n, there is a single problem in the SVA problem distribution on n agents, runtimes of the IP and IDP-IP algorithm on problems from the SVA distribution depend only on the number of agents. Hence, the only variability in the algorithms' runtime is any miscellaneous processor activity on the computer while it runs.

The time required to guarantee particular approximation ratios for each of the algorithms on problems from the *SVA* distribution ranging from 13 to 17 agents is presented in Figure V.10. In particular, Figure V.10 presents a graph of:

1. the time required by the IP algorithm to guarantee a $\frac{1}{4}$-approximate solution,
2. the time required by Sandholm's algorithm to guarantee $\frac{1}{2}$- and $\frac{1}{3}$-approximate solutions,
3. the time required by Dang and Jenning's algorithm to guarantee a $\frac{1}{3}$-approximate solution,
4. the time required by the presented deterministic approximation algorithm to guarantee a $\frac{2}{3}$-approximate solution.

The deterministic approximation algorithm is capable of generating a $\frac{2}{3}$-approximate solution in significantly less time than required by the other algorithms, even though it pro-

vides a better solution quality guarantee. Dang and Jenning's algorithm guarantee's a $\frac{1}{3}$-approximate solution more quickly than Sandholm's algorithm and is significantly faster than the IP algorithm's generation of a $\frac{1}{4}$-approximate solution. Finally, Sandholm's algorithm guarantees a $\frac{1}{2}$-approximate solution in less time than required by the IP algorithm to guarantee a $\frac{1}{4}$-approximate solution. The results presented in Figure V.10 match the theoretical results presented in Corollary 17 and demonstrate that the SVA problem distribution is a difficult problem for the IP algorithm.

Figure V.10: Comparison of the runtimes of the IP algorithm, Sandholm's algorithm, Dang and Jenning's algorithm, and the deterministic $\frac{2}{3}$-approximation algorithm on problems from the SVA distribution consisting of 13 to 17 agents.

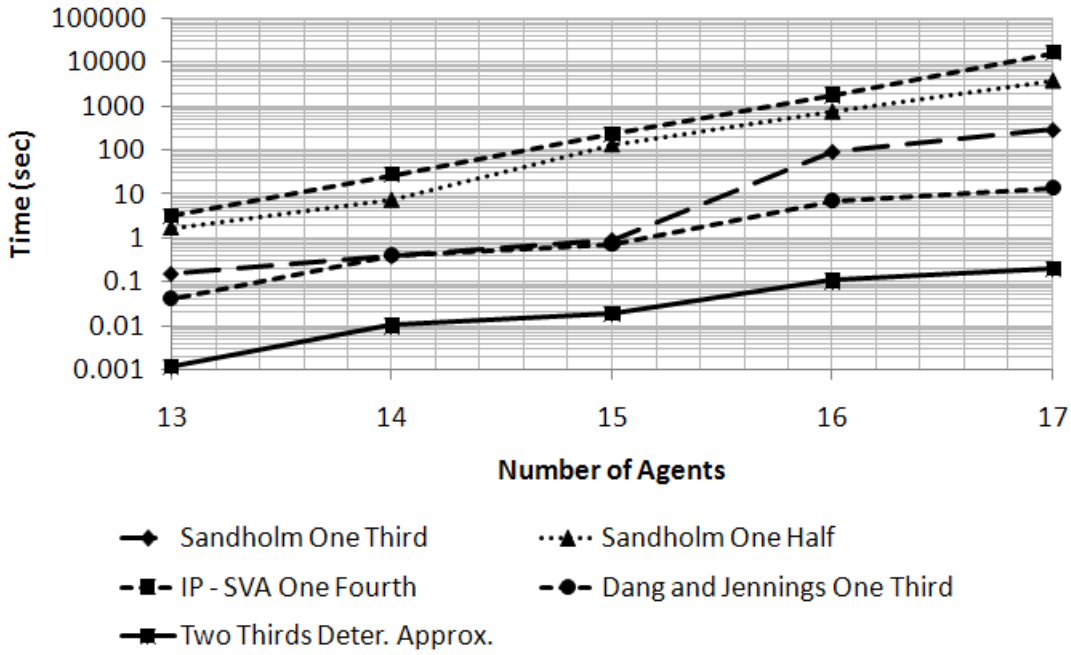
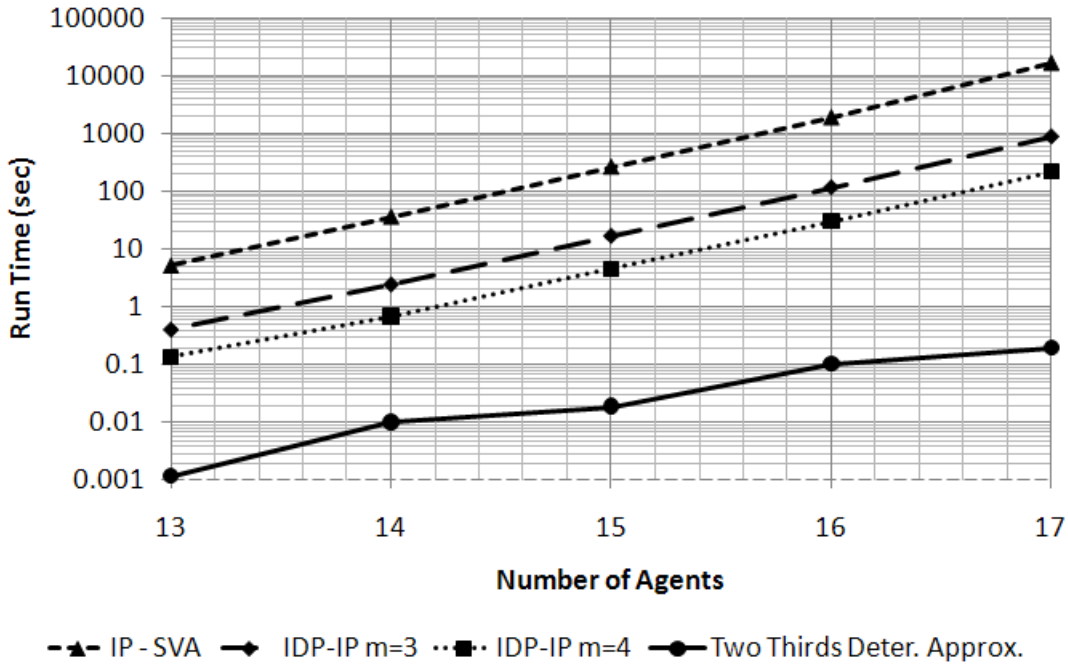


Figure V.11 shows the time required by the IP algorithm and the IDP-IP algorithm (for various values of the parameter m) to guarantee a $\frac{1}{2}$-approximate solution and the time required by the deterministic $\frac{2}{3}$-approximation algorithm on problems from the *SVA* distribution ranging from 13 to 17 agents. The empirical results confirm the theoretical analysis. While the IDP-IP algorithm outperforms the IP algorithm, it still requires significantly longer than the deterministic $\frac{2}{3}$-approximation algorithm, even to provide lower

approximation guarantee.

Figure V.11: Comparison of the runtimes of the IP algorithm, IDP-IP algorithm, and there deterministic $\frac{2}{3}$-approximation algorithm on problems from the SVA distribution on problems consisting of between 13 to 17 agents.

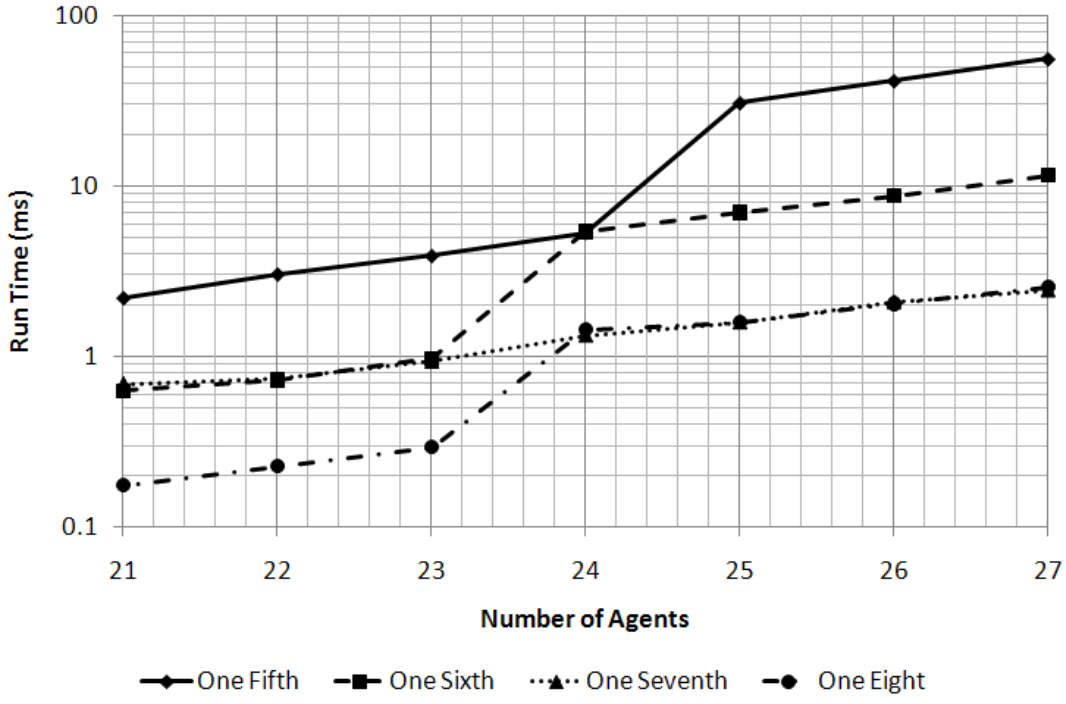


V.3 Approximate Coalition Structure Generation in Monotonic Games

Graphs of the runtime of the monotonic $\frac{1}{5}$-, $\frac{1}{6}$-, $\frac{1}{7}$-, and $\frac{1}{8}$-approximation algorithms is provided in Figure V.12. The approximation algorithms ran extremely quickly in all cases, even for problems consisting of 27 agents. The maximum average runtime was less than 56 ms.

The runtimes of the different approximation algorithms coincided in many cases. This is a result of the fact that the monotonic $\frac{1}{r}$-approximation algorithm simply needs to solve all subproblems consisting of at most $\lfloor \frac{n}{r} \rfloor$ agents to guarantee a $\frac{1}{r}$-approximate solution. Taking the floor of $\frac{1}{r}$ causes the runtimes for the different approximation guarantees to sometimes coincide. For example, $\lfloor \frac{24}{6} \rfloor = \lfloor \frac{24}{5} \rfloor$, thus the $\frac{1}{5}$-approximation algorithm and the $\frac{1}{6}$-approximation algorithm have the same runtime on problems consisting of 24 agents.

Figure V.12: Comparison of the runtimes of the monotonic approximation algorithm for the $\frac{1}{5}$-, $\frac{1}{6}$-, $\frac{1}{7}$-, and $\frac{1}{8}$-approximation guarantees on problems consisting of 21 to 27 agents.



V.4 Discussion

This chapter has presented empirical results comparing the approximation algorithms for coalition structure generation to existing algorithms from the literature. It was shown that the presented approximation algorithms are capable of generating better performance guarantees in less time then the algorithms of Sandholm et al. (Sandholm et al., 1999) and Dang and Jennings (Dang and Jennings, 2004).

The current state-of-the-art IP algorithm out performs the presented determinisitic approximation algorithm on four standard problem distributions defined in the literature. The IP algorithm is capable of generating relatively high quality bounds on the value of solutions in each subspace for each of the four problem distributions. A contribution of this chapter is to present the Single Valuable Agent (SVA) problem distribution. The SVA is the first problem distribution on which the IP algorithm is unable to generate quality subspace solution bounds. The presented deterministic approximation algorithm, Sandholm's

algorithm, and Dang and Jenning's algorithm all outperformed the IP algorithm in terms of the time required to guarantee a given approximation ratio on problems from the SVA distribution.

The SVA problem is difficult for the IP algorithm to generate quality subspace bounds. For each coalition size, there are coalitions of very high value (compared to most other coalitions); however, every coalition structure contains only one such high valued coalition. This observation implies that any coalitional game, can be transformed into an equivalent game (from a coalition structure generation point of view) on which the IP algorithm is unable to generate good upper bounds on the values of coalition structures within each subspace. Given any coalitional game, (N, v), define a new game (N, v'), where

$$v'(C) = \begin{cases} v(C) + M & \text{if } i \in C \\ v(C) & \text{if } otherwise, \end{cases}$$

and M is any large value. Notice that both coalitonal games have the same set of optimal coalition structures. v' shares many of the characteristics of the SVA distribution. That is, if M is sufficiently large, then the bounds generated by the IP algorithm are exaggerated. The observation that any coalitional game can be transformed into a new, equivalent game under which the IP algorithm is unable generate good bounds begs the question as to whether the reverse is true. That is, is it possible to transform the original given game into an equivalent game under which good bound estimates can be made. Stated another way: Is it possible to "normalize" the values of a coalitional game so that quality upper bounds can be inferred? This is an interesting line of future work.

This concludes the study of approximation algorithm for coalition structure generation. The next chapter explores the coalition structure generation problem in situations where there is only a small number of distinct agent types.

CHAPTER VI

Bounded Agent Types

This chapter explores the coalition structure generation problem in coalitional games where the number of distinct agent types is small. Informally, two agents are of the same type if they are indistinguishable from one another. Coalitional games with few agent types is a natural situation to consider, as in many realistic settings, the number of distinct agents is small compared to the total number of agents. For example, in situations where agents are physical robots, it is likely that many of the robots are identical.

Two agents are of the same type if they are somehow indistinguishable (Shrot et al., 2010; Ueda et al., 2011a,b; Aziz and de Keijzer, 2011). Two definitions of agent equivalence are employed. The most general definition is that of strategic equivalence.

Definition 39 (strategic equivalence)**.** *Two agents $i, j \in N$ are* strategically equivalent*, $i \sim j$, if for all $C \subseteq N \setminus \{i, j\}$, $v(C \cup \{i\}) = v(C \cup \{j\})$. If i and j are strategically equivalent, they are said to be of the same type.*

In some cases, it may be difficult to determine if two agents are strategically equivalent. For example, determining if two agents are strategically equivalent in a weighted voting game is *co-NP-hard* (Aziz and de Keijzer, 2011). However, a stronger notion of equivalence can be made with respect to a given coalitional game representation (Shrot et al., 2010). Shrot et al. (Shrot et al., 2010) observe that in any compact representation of a coalitional game, each agent is associated with an identifier (e.g., a node in a graph game). Shrot et al. define two agents to be representationally equivalent if they differ only in their identifier.

Definition 40 (representational equivalence)**.** *Let $i, j \in N$ be two agents in a compactly representable coalitional game G. Let $R(G)$ be the representation of the game and let id_i and*

id_j be the identifiers of i and j, respectively, in $R(G)$. Agents i and j are representationally equivalent*, $i \sim_r j$, if swapping their identifiers results in the same representation, $R(G)$.*

The precise definition of what it means for two agents i and j to be representationally equivalent varies from game to game. However, in all cases, if i and j are representationally equivalent, then i and j are strategically equivalent.

It is easy to see that strategic and representational equivalence are equivalence relations on N. For a coalitional game (N, v) and $i \in N$, let $[i]$ be the equivalence class of the type of i. It will be clear from context whether the underlying equivalence relation is strategic or representational equivalence. Two coalitions $C, S \subseteq N$ are strategically (representationally) equivalent if for each $i \in N$, $|[i] \cap C| = |[i] \cap S|$. That is, C and S are equivalent if they contain equal numbers of members from each equivalence class. Let $[C]$ be the equivalence class of a coalition $C \subseteq N$. Define the ordering $\leq$ coalition equivalence classes as: $[S] \leq [C]$ if there exists $C' \in [C]$ such that $S \subseteq C'$. Finally, for $[S] \leq [C]$ define $[C] - [S]$ as $[C' \setminus S]$, where $S \subseteq C' \in [C]$.

A number of computational problems become polynomial time solvable in coalitional games when there is a constant number of agent types. For example, determining if a given imputation is in the core, determining if the core is empty, and computing the Shapley value can all be achieved in polynomial time when the number of agent types is constant (Ueda et al., 2011a,b). The majority of such results stem from the observation that it is sufficient to consider only coalition equivalence classes and not individual coalitions.

VI.1 A General Algorithm

An optimal coalition structure can be found quickly in coalitional games with few agent types. Algorithm 8 is a modification of the standard dynamic programming algorithm for coalition structure generation and is based on the observation that it is sufficient to restrict our attention to the equivalence classes of coalitions, rather than all coalitions. Algorithm 8 is applicable to both games with bounded strategic or bounded representational types. The

essence of Algorithm 8 has been rediscovered a number of times (Aziz and de Keijzer, 2011; Service and Adams, 2011a; Ueda et al., 2011a,b)

Algorithm 8 Coalition Structure Generation For Bounded Agent Types

```
for [C] ≤ [N] do
    V([C]) = v(C)
    for [S] ≤ [C] do
        if v([S]) + V([C] − [S]) > V([C]) then
            V([C]) = v([S]) + V([C] − [S])
        end if
    end for
end for
```

Theorem 18. *Algorithm 8 runs in $O(n^{2k})$, where k is the number of agent types.*

Proof. There are $O(n^k)$ coalition types and for each, the maximum over at most $O(n^k)$ values must be taken. □

As shown, Algorithm 8 computes only the value of the optimal coalition structure. However, it may be easily adapted to find an optimal coalition structure by storing, for each coalition equivalence class $[C]$, pointers to the two subclasses $[S] < [C]$ and $[C] - [S]$ that maximize their value (or storing a single pointer to $[C]$, if $[C]$ is itself an optimal partition). As in the case with the standard dynamic programming algorithm, an optimal coalition structure can be identified by walking along the stored pointers. Traversing the points can be accomplished in $O(n)$ time, as there are at most n coalitions in any coalition structure. This process does not immediately result in a coalition structure, rather it results in a set of coalition equivalence classes. A coalition structure can be constructed given a set of coalition equivalence classes K as follows. Maintain the current set of agents A and for each agent type a list of all agents of that type. Process each coalition class $[C] \in K$ in an arbitrary order. For each $[C] \in K$, construct a coalition $C' \in [C]$, by selecting an appropriate number of agents of each type using the agent type lists. Remove all agents appearing in C' from A and the agent type lists and proceed to the next coalition equivalence class in K. Since, the value of C' depends only upon the number of each type of agent that appears

in C', and not on the particular agents in C', this process results in an optimal coalition structure. Hence, an optimal coalition structure can be constructed in $O(n^2)$ time, since there are at most n coalitions and each can be constructed in $O(n)$ time. Therefore, an optimal coalition structure can be found in $O(n^{2k})$ time when there are k agent types.

VI.2 Classes of Coalitional Games

The section considers the coalition structure generation problem in two subclasses of coalitional games with bounded agent types. It is shown that in both subclasses, the coalition structure generation problem is fixed parameter tractable in the number of agent types.

Throughout the remainder of this chapter, $poly(n)$ is used to represent an arbitrary polynomial in n. More generally, $poly(n_1, \cdots, n_k)$ is used to represent an arbitrary polynomial in $n_1, \cdots, n_k$. Recall that an algorithm is fixed parameter tractable in the number of agent types, k, if its runtime is $O(f(k)poly(n))$, where $f(k)$ is an arbitrary function of k and n is the size of the input.

The first subclass consists of those games in which it is known that there exists an optimal coalition structure where all agents of the same type appear in the same coalition.

Theorem 19. *Let (N, v) be a coalitional game with k types. If (N, v) has an optimal coalition structure in which all agents of the same type appear in the same coalition, then an optimal coalition structure can be found in $O(3^k poly(n))$ time.*

Proof. Let $T = \{t_1, \cdots, t_k\}$ be the set of agent types in N. Identify each agent equivalence class, $[a]$, with its corresponding type in T. For each $S \subseteq T$, let $T(S) = \{a \in N : [a] \in S\}$ be the set of agents whose type appears in S. Consider the following coalitional game (T, v') defined as, for $S \subseteq T$, $v'(S) = v(T(S))$. By assumption, there is an optimal coalition structure in (N, v) such that every agent of the same type appears in the same coalition. Hence, any optimal coalition structure in (T, v') has value equal to the value of the optimal coalition structures in (N, v). Given an optimal coalition structure $CS = \{C_1, \cdots, C_m\}$ in (T, v'), $\{T(C_1), \cdots, T(C_m)\}$ is an optimal coalition structure in (N, v).

Since $|T| = k$, an optimal coalition structure can be found in $O^*(3^k) = O(3^k poly(n))$ time in (T, v'), since $k \leq n$. This coalition structure can be converted to an optimal coalition structure in (N, v) in time polynomial in n. □

The second subclass considered represents the opposite extreme. Assume it is known that there exists an optimal coalition structure where all agents of the same type appear in different coalitions. We begin with a straightforward improvement over Algorithm 8.

Theorem 20. *If there exists an optimal coalition structure in which all agents of the same type appear in different coalitions, then an optimal coalition structure can be found in* $O(2^k n^k)$ *time.*

Proof. An optimal coalition structure can be obtained in $O(2^k n^k)$ time by employing the following variant of Algorithm 8. For a coalition type $[C]$, when calculating the value of $V([C])$, it suffices to have the *for* loop on line three loop over only those coalition types that contain at most one agent of each type. Since there exists an optimal coalition structure in which each agent of the same representational type appears in a different coalition, this modification to Algorithm 8 ensures that an optimal solution is found.

Since there are $O(n^k)$ coalition types and for each coalition type the maximum over $O(2^k)$ values must be taken, the total runtime of the modified Algorithm 8 is $O(2^k n^k)$. □

While the preceding Theorem provides an improvement over Algorithm 8, the coalition structure generation problem is in fact fixed parameter tractable in games where it is known that there exists an optimal coalition structure in which all agents of the same type appear in distinct coalitions.

Theorem 21. *Let* (N, v) *be a coalitional game with k types. If* (N, v) *has an optimal coalition structure in which all agents of the same type appear in different coalitions, then an optimal coalition structure can be found in* $O(4^{k^2} poly(n))$ *time.*

The proof of Theorem 21 requires the following Lemma.

Lemma 10. *Let $y_1, \cdots, y_m \in \{0,1\}^k$ and let $n = y_1 + \cdots + y_m$. There exists a partition of $\{y_1, \cdots, y_m\}$ into sets A and B such that:*

1. $|B| - 1 \leq |A| \leq |B|$ *and*

2. $|A(i) - B(i)| \leq 2^{i-1}$ *for all* $i = 1, \cdots, k$,

where $A(i) = \sum_{y \in A} y(i)$ and $y(i)$ is the i-th component of the vector y.

Proof. We proceed by induction on k. The lemma is readily verified when $k = 1$. Assume that the result holds for all vectors of length $k-1$. Let $Y_l = \{y_i | y_i(1) = l\}$ for $l \in \{0,1\}$. Since the vectors in Y_l all agree on the first coordinate, we may treat them as vectors of length $k-1$ by considering only the remaining $k-1$ coordinates. Hence, Y_l may be partitioned into A_l and B_l, such that:

1. $|B_l| - 1 \leq |A_l| \leq |B_l|$ and

2. $|A_l(i) - B_l(i)| \leq 2^{i-2}$ for all $i = 2, \cdots, k$.

Notice that condition 2 is offset by 1 due to the vectors in both Y_0 and Y_1 having identical first coordinates.

Now consider the partition of $y_1, \cdots, y_m$ into $A = A_0 \cup B_1$ and $B = A_1 \cup B_0$. Assume, without loss of generality, that $|A| \leq |B|$ (otherwise swap the sets A and B). First notice that these two sets differ in size by at most 1. For any $i = 2, \cdots, k$ we have

$$\begin{aligned} |A(i) - B(i)| &= |A_0(i) + B_1(i) - A_1(i) - B_0(i)| \\ &= |A_0(i) - B_0(i) + B_1(i) - A_1(i)| \\ &\leq |A_0(i) - B_0(i)| + |B_1(i) - A_1(i)| \\ &\leq 2^{i-2} + 2^{i-2} = 2^{i-1}. \end{aligned}$$

Finally:

$$|A(1)-B(1)| \quad = \quad |0+B_1(1)-A_1(1)-0| = |B_1(1)-A_1(1)| \leq 1,$$

since the sets A_1 and B_1 differ in size by at most 1. □

Proof of Theorem 21. Let CS be a coalition structure such that each $C \in CS$ contains at most one of each agent type. The coalitions in CS may be regarded as vectors over $\{0,1\}^k$, for k types, indicating which agent types are present in the coalition. Lemma 10 implies that the coalitions in CS can be partitioned into two sets A and B, such that

1. $||A|-|B|| \leq 1$ and
2. $|A(i)-B(i)| \leq 2^{i-1} \leq 2^{k-1}$ for all $i = 1,\cdots,k$,

where $A(i)$ is the number of agents of type i that appear in coalitions in A. Hence, in Algorithm 8, when computing $V([C])$, it suffices to consider the partitions of $[C]$ into A and B such that for each agent type i, the number of agents of type i in A and B differ by at most 2^{k-1}. Then, if there are $A(i)$ agents of type i in A, there are $C(i)-A(i)$ agents of type i in B. Hence, $\frac{C(i)}{2}-2^{k-2} \leq A(i) \leq \frac{C(i)}{2}+2^{k-2}$. Therefore, when computing $V([C])$, it suffices to take the maximum over

$$\prod_{i=1}^{k}\left[\left(\frac{C(i)}{2}+2^{k-2}\right)-\left(\frac{C(i)}{2}-2^{k-2}\right)\right] \quad = \quad \prod_{i=1}^{k}\left[2^{k-1}\right] \leq 2^{k^2},$$

values, one for each possible partition.

Now, let N be the set of agents and let n_i be the number of agents in N of type i. Notice that to compute the value of $V([N])$, we require only the values of the optimal solutions on the subproblems consisting of between $\frac{n_i}{2}-2^{k-2}$ and $\frac{n_i}{2}+2^{k-2}$ agents of type i, for each $i = 1,\cdots,k$. Similarly, when computing the optimal solution to a subproblem consisting

of n'_i agents of type i (for each i), with $\frac{n_i}{2} - 2^{k-2} \leq n'_i \leq \frac{n_i}{2} + 2^{k-2}$, only the solutions to subproblems consisting of between $\frac{n_i}{4} - \frac{2^{k-2}}{2} - 2^{k-2}$ and $\frac{n_i}{2} + \frac{2^{k-2}}{2} + 2^{k-2}$ agents of type i are required. Continuing in this fashion, it is necessary to only compute the optimal solutions to the subproblems $n'_1, \cdots, n'_i, \cdots, n'_k$ that, for each agent type i, satisfy:

$$\begin{aligned}
\frac{n_i}{2} - 2^{k-2} \leq \quad & n'_i \quad \leq \frac{n_i}{2} + 2^{k-2} \\
\frac{n_i}{4} - \frac{2^{k-2}}{2} - 2^{k-2} \leq \quad & n'_i \quad \leq \frac{n_i}{4} + \frac{2^{k-2}}{2} + 2^{k-2} \\
\frac{n_i}{8} - \frac{2^{k-2}}{4} - \frac{2^{k-2}}{2} - 2^{k-2} \leq \quad & n'_i \quad \leq \frac{n_i}{8} + \frac{2^{k-2}}{4} + \frac{2^{k-2}}{2} + 2^{k-2} \\
& \vdots
\end{aligned}$$

Since $\sum_{i \in \mathbb{N}} \frac{2^{k-2}}{2^i} = 2^{k-1}$, it is sufficient to solve only those subproblems of the form $\frac{n_i}{2^l} - 2^{k-1} \leq n'_i \leq \frac{n_i}{2^l} + 2^{k-1}$. Notice that each "level" of subproblems consists of $O(2^{k^2})$ table entries that must be computed and there are $O(\log(n))$ "levels". Therefore, there are $O(\log(n)2^{k^2})$ table entries to compute and computing the value of each entry in the table requires taking the maximum over $O(2^{k^2})$ values. Thus, an optimal coalition structure, where each agent of a given type appears in different coalitions, can be found in $O(4^{k^2} poly(n))$ time. □

The algorithm presented in the proof of Theorem 21 can be implemented by maintaining $O(log(n))$ tables each of $O(2^{k^2})$ entries, rather than a single larger table.

VI.3 Coalitional Skill Games

This section considers the computational complexity of several natural problems in coalitional skill games with bounded representational types. Recall that a coalitional skill game consists of set of agents N, a set of skills S, a set of tasks T, and an increasing utility function $u : 2^T \to \mathbb{R}^{\geq 0}$. For the results presented in this section, we assume that querying u can

be done in $O(poly(n,|S|,|T|))$ time[1]. Thus, we can compute the value of any coalition in polynomial time.

We begin by defining representational equivalence in coalitional skill games.

Definition 41. *In a coalitional skill game, two agents i and j are representationally equivalent if they possess the same set of skills (i.e., $S(i) = S(j)$).*

Notice that if the number of skills is $k = |S|$, then there are most 2^k representational types. Therefore, every result in this section stating that a given computational problem is fixed parameter tractable in the number of representational types has an immediate corollary stating that the same problem is fixed parameter tractable in the number of skills.

Example 22 shows that agents may be strategically equivalent without being representationally equivalent in coalitional skill games.

Example 22. *Consider the coalitional skill game with $S = \{s_i, s_j\}$, $T = \{t_i, t_j\}$ and $N = \{i, j\}$. Let $u(\{t_i\}) = u(\{t_j\}) = 1$ and let $u(\{t_i, t_j\}) = 2$. Task t_i requires only skill s_i and agent i possess only skill s_i. Similarly, task t_j requires only skill s_j and agent j possess only skill s_j. Clearly i and j are not representationally equivalent; however, they are strategically equivalent as $v(\emptyset \cup \{i\}) = v(\emptyset \cup \{j\})$.*

The following lemma provides the basis for many of the results regrading coalitional skill games with bounded representational types.

Lemma 11. *In any coalitional skill game, there exists an optimal coalition structure in which no two agents of the same representational type appear in the same coalition.*

Proof. Let OPT be an optimal solution and let i and j be two representationally equivalent agents that appear in the same coalition $C \in OPT$. Since i and j are representationally

[1] Alternatively, all the results can be interpreted as applying to subclasses of coalitional skill games where u is always compactly representable and queries to u can be answered in polynomial time.

equivalent, $S(i) = S(j)$. Note that

$$\begin{aligned} S(C) &= \bigcup_{k \in C} S(k) \\ &= \bigcup_{k \in C \setminus \{j\}} S(k) \\ &= S(C \setminus \{j\}), \end{aligned}$$

where the second to last equality follows, since $i \in C \setminus \{j\}$ and $S(i) = S(j)$. Hence, $v(C) = v(C \setminus \{j\})$, as the value of a coalition depends only on the skills its members possess. A new coalition structure OPT' can be formed by removing j from C in OPT and adding in the singleton coalition $\{j\}$. The newly formed coalition structure OPT' is also optimal.

Continuing in this fashion, an optimal coalition structure can be created in which every agent of the same representational type appears in different coalitions. □

The next two corollaries follow immediately from the preceding lemma and Theorem 21.

Corollary 23. *An optimal coalition structure in coalitional skill games with k representational types can be found in $O(4^{k^2} poly(n, |S|, |T|))$ time.*

Corollary 24. *An optimal coalition structure in coalitional skill games with k skills can be found in $O(4^{4^k} poly(n, |S|, |T|))$ time.*

Computational problems related to the *core* and Banzhaf index in coalitional skill games with bounded representational types are now considered. Recall that an imputation p is blocked by a coalition C, if $p(C) < v(C)$. That is, if C can earn more on its own, then it is awarded under p.

Lemma 12. *Consider a coalitional skill game (N, v) with k representational types. If p is an imputation that is not in the core, then p is blocked by a coalition that contains at most one agent of each type.*

Proof. If, for some agent $i \in N$, $p_i < 0$, then p_i is blocked by a singleton coalition and the lemma holds. Thus, assume that $p_i \geq 0$ for all $i \in N$.

Since p is not in $core(v)$, it is blocked by some coalition C. Assume that there exists $i, j \in C$ such that the representational type of i and j are the same. Since $i \sim_r j$, $S(i) = S(j)$. Hence, since $i \in C \setminus \{j\}$, $S(C \setminus \{j\}) = S(C)$ and thus, $v(C \setminus \{j\}) = v(C)$. Since $p(C \setminus \{j\}) \leq p(C)$, $C \setminus \{j\}$ blocks p. Continuing in this fashion results in a coalition that blocks p and contains at most one agent of each type. □

Corollary 25 employs the preceding lemma to show that determining if a given imputation is in the core is fixed parameter tractable in coalitional skill games.

Corollary 25. *Given a coalitional skill game with k representational types and an imputation p, it can be determined if p is in core*(v) *in* $O(2^k poly(n))$ *time.*

Proof. By the preceding lemma, to verify that p is in $core(v)$, it suffices to verify that p is not blocked by any coalition containing at most one agent of each type. Let T be the set of agent types. For each agent type $t \in T$, let a_t be an agent of type t such that p_{a_t} is minimized. Assume that p is blocked by a coalition C containing at most one agent of each type. Let S be the set of types of agents in C and $C' = \{a_t | t \in S\}$. By the choice of the agents a_t, $p(C') \leq p(C)$; however, $v(C) = v(C')$. Therefore, p is blocked by C' as well. Thus, it suffices to verify that p is not blocked by any coalition consisting of agents from the set $\{a_t | t \in T\}$. Since there are only 2^k such coalitions, a blocking coalition can be found, it if exists, in $O(2^k poly(n))$ time. □

The following theorem shows that the Banzhaf index can be computed efficiently in coalitional skill games with few representational types.

Theorem 26. *The Banzhaf index can be computed in* $O(2^k poly(n))$ *time.*

Proof. For each representational agent type t, let n_t be the number of agents of type t and let T be the set of agent types. Finally, let t_i be the type of agent i. For a set of agent types

$S \subseteq T$, let $v'(S)$ be the value of a coalition that contains at least one agent of each type in S and no agents of a type $t \notin S$. Let $t : 2^N \to 2^T$ be the map that sends a coalition C to the set of agent types that appear in C.

The Banzhaf index of an agent i is

$$\phi_i(v) = \frac{1}{2^{n-1}} \sum_{C \subseteq N \setminus \{i\}} \left(v(C \cup \{i\}) - v(C) \right).$$

Notice that the marginal contribution of an agent i to a coalition C depends only on which representational types of agents are in C and not on the quantity of each type in C. Also notice that if C already contains an agent of the same representational type as i, then the marginal contribution of i to C is 0 (i.e., $v(C \cup \{i\}) - v(C) = 0$).

We claim that the Banzhaf index of an agent i can be expressed as

$$\phi_i(v) = \frac{1}{2^{n-1}} \sum_{S \subseteq T \setminus \{t_i\}} \left(\left[\prod_{t \in S} (2^{n_t} - 1) \right] \left(v'(S \cup \{t_i\}) - v'(S) \right) \right). \tag{VI.1}$$

Begin by noting that Equation VI.1 can be computed in time $O(2^k poly(n))$.

The marginal contribution of agent i to any coalition not containing i is counted exactly once in the sum in Equation VI.1. Let $C \subseteq N \setminus \{i\}$ be an arbitrary coalition. If C contains an agent of type t_i, then the marginal contribution of i to C is 0 and is trivially counted in the sum. Now consider the case where C contains no agents of type t_i. Let S be the set of agent types that appear in C (i.e., $t(C) = S$). The marginal contribution of i to C is $v'(S \cup \{t_i\}) - v'(S)$. Therefore, as $v'(S \cup \{t_i\}) - v'(S)$ contributes to the sum $\prod_{t \in S} (2^{n_t} - 1)$ times, it suffices to show that there are precisely $\prod_{t \in S} (2^{n_t} - 1)$ coalitions C, such that $t(C) = S$. Note that $t(C) = S$ if and only if for each type $t \in S$, there is at least one agent of type t in C. For each type $t \in S$, there are $2^{n_t} - 1$ nonempty subsets of agents of type t. Finally, notice that we may select a nonempty subset of agents of type t independently for each type $t \in S$. Therefore the number of coalitions C, such that $t(C) = S$ is $\prod_{t \in S} (2^{n_t} - 1)$. □

VI.4 Graph Games

This subsection considers a restricted case of graph games with a bounded number of types. The following results show that representational and strategic types coincide in graph games.

Theorem 27. *In a graph game $G = (V,E,w)$, $i,j \in V$ are the same strategic type if and only if for all $k \in V \setminus \{i,j\}$, $w(\{i,k\}) = w(\{j,k\})$.*

Proof. Let v be the coalitional game defined by G.

Certainly, if i and j are of the same type, then for every $k \in V \setminus \{i,j\}$, $w(\{i,k\}) = v(\{i,k\}) = v(\{j,k\}) = w(\{j,k\})$.

Conversely, if for all $k \in V \setminus \{i,j\}$, $w(\{i,k\}) = w(\{j,k\})$, then for any coalition $C \subseteq N \setminus \{i,j\}$,

$$\begin{aligned} v(C \cup \{i\}) &= v(C) + \sum_{k \in C} w(\{i,k\}) \\ &= v(C) + \sum_{k \in C} w(\{j,k\}) \\ &= v(C \cup \{j\}). \end{aligned}$$

□

Corollary 28. *In a graph game $G = (V,E,w)$, two agents i and j are strategically equivalent if and only if they are representationally equivalent.*

Proof. The corollary follows immediately from Theorem 27, as two agents i and j are representatively equivalent if and only if for every third agent k, $w(\{i,k\}) = w(\{j,k\})$. □

Determining if two agents are of the same type can be computed in polynomial time for graph games.

Theorem 29. *In a graph game $G = (V,E,w)$, the partition of agents into their respective types can be found in $O(n^3)$ time.*

Proof. Determining if two agents, i and j, are of the same type can be accomplished in $O(n)$ time by comparing the weights on the edges between agents i and k and agents j and k for every other agent k. Since there are $O(n^2)$ pairs of vertices, the theorem follows. □

The remainder of this section considers only graph games that satisfy the following property: $(*)$ If i and j are two agents of the same type in a graph game $G = (V,E,w)$, then $w(\{i,j\}) \geq 0$.

Theorem 30. *If $G = (V,E,w)$ is a graph game that satisfies $(*)$, then there is an optimal coalition structure in which all agents of the same type belong to the same coalition.*

Proof. Let CS be any coalition structure and let $i, j \in N$ be two agents of the same type, such that $i \in C_i \neq C_j \ni j$, where $C_i, C_j \in CS$. The value of coalition C_i is

$$\sum_{k,l \in C_i \setminus [i]} w(\{k,l\}) + |C_i \cap [i]| \sum_{k \in C_i \setminus [i]} w(\{i,k\}) + w(\{i,j\}) \binom{|C_i \cap [i]|}{2}.$$

Likewise, the value of C_j is

$$\sum_{k,l \in C_j \setminus [i]} w(\{k,l\}) + |C_j \cap [i]| \sum_{k \in C_j \setminus [i]} w(\{i,k\}) + w(\{i,j\}) \binom{|C_j \cap [i]|}{2}.$$

Assume, without loss of generality that for each $j \in [i]$,

$$\sum_{k \in C_i \setminus [i]} w(\{i,k\}) \geq \sum_{k \in C_j \setminus [i]} w(\{i,k\}).$$

Define coalitions C_i' and C_j' as follows:

$$\begin{aligned} C_i' &= C_i \cup \left(C_j \cap [i]\right), \\ C_j' &= C_j \setminus [i]. \end{aligned}$$

The value of coalition C'_i is

$$\begin{aligned}
& \sum_{k,l \in C'_i \setminus [i]} w(\{k,l\}) + |C'_i \cap [i]| \sum_{k \in C'_i \setminus [i]} w(\{i,k\}) + w(\{i,j\}) \binom{|C'_i \cap [i]| - 1}{2} \\
= & \sum_{k,l \in C_i \setminus [i]} w(\{k,l\}) + \big(|C_i \cap [i]| + |C_j \cap [i]|\big) \sum_{k \in C_i \setminus [i]} w(\{i,k\}) \\
& + w(\{i,j\}) \binom{|C_i \cap [i]| + |C_j \cap [i]|}{2} \\
> & \sum_{k,l \in C_i \setminus [i]} w(\{k,l\}) + \big(|C_i \cap [i]| + |C_j \cap [i]|\big) \sum_{k \in C_i \setminus [i]} w(\{i,k\}) \\
& + w(\{i,j\}) \left(\binom{|C_i \cap [i]|}{2} + \binom{|C_j \cap [i]|}{2} \right).
\end{aligned}$$

The value of C'_j is

$$\sum_{k,l \in C'_j} w(\{k,l\}) = \sum_{k,l \in C_j \setminus [i]} w(\{k,l\}).$$

Therefore, we have that:

$$\begin{aligned}
v(C'_i) + v(C'_j) > & \sum_{k,l \in C_i \setminus [i]} w(\{k,l\}) + \big(|C_i \cap [i]| + |C_j \cap [i]|\big) \sum_{k \in C_i \setminus [i]} w(\{i,k\}) \\
& + w(\{i,j\}) \left(\binom{|C_i \cap [i]|}{2} + \binom{|C_j \cap [i]|}{2} \right) + \sum_{k,l \in C_j \setminus [i]} w(\{k,l\}) \\
= & \sum_{k,l \in C_i \setminus [i]} w(\{k,l\}) + |C_i \cap [i]| \sum_{k \in C_i \setminus [i]} w(\{i,k\}) + w(\{i,j\}) \binom{|C_i \cap [i]|}{2} \\
& + \sum_{k,l \in C_j \setminus [i]} w(\{k,l\}) + |C_j \cap [i]| \sum_{k \in C_j \setminus [i]} w(\{i,k\}) + w(\{i,j\}) \binom{|C_j \cap [i]|}{2} \\
= & v(C_i) + v(C_j).
\end{aligned}$$

□

The following corollary follows directly from Theorems 19 and 30.

Corollary 31. *If a graph game $G = (V, E, w)$ satisfies $(*)$, then an optimal coalition structure can be found in $O(3^k poly(n))$ time.*

VI.5 Marginal Contribution Networks

This section considers the coalition structure generation problem in a subclass of marginal contribution networks with bounded representational types. The first example shows that the notions of representation and strategic equivalence are different in marginal contribution networks.

Example 32. *Consider the marginal contribution network defined by the set of rules* $R = \{r_i, r_j\}$ *and two agents* $N = \{i, j\}$. *Let* $r_i = (P_{r_i}, N_{r_i}, v_{r_i})$, *where* $P_{r_i} = \{i\}$ *and* $N_{r_i} = \{j\}$ *and* $v_{r_i} = 1$. *Let* r_j *be defined similarly as* $P_{r_j} = \{j\}$ *and* $N_{r_i} = \emptyset$ *and* $v_{r_j} = 1$. *Clearly* i *and* j *are not representationally equivalent; however, they are strategically equivalent as* $v(\emptyset \cup \{i\}) = v(\emptyset \cup \{j\})$.

This section considers marginal contribution networks that satisfy the following property: $(**)$ If i and j are two representationally equivalent agents, then for each rule $R = (P_r, N_r, v_r)$, either both i and j appear in P_r, both appear in N_r, or neither appear in $P_r \cup N_r$.

Lemma 13. *In a marginal contribution network that satisfies* $(**)$*, there exists an optimal coalition structure in which each agent of the same representational type appears in the same coalition.*

Proof. Let OPT be an optimal solution and let i and j be two representationally equivalent agents that appear in different coalitions $C_i, C_j \in OPT$, respectively.

Let $r = (P_r, N_r, v_r)$ be a rule that applies to C_i. This means that $P_r \subseteq C_i$ and $N_r \subseteq N \setminus C_i$. Hence, $i \notin N_r$, since $i \in C_i$. By $(**)$, $j \notin N_r$. Similarly, as $j \notin P_r \subseteq C_i$, $i \notin P_r$. Thus, the rule r applies to $C_i \setminus \{i\}$.

Now consider the coalition $S = C_j \cup \{i\}$. Let $r = (P_r, N_r, v_r)$ be a rule that applies to C_j. Thus, $P_r \subseteq C_j \subseteq S$ and $N_r \subseteq N \setminus C_j$. Assume that $N_r \not\subseteq N \setminus S$. Then, $i \in N_r$. However, since i and j are representationally equivalent, j must be in N_r as well. This is a contradiction, as $N_r \subseteq N \setminus C_j$ and $j \in C_j$. Therefore, r applies to $S = C_j \cup \{i\}$ as well.

Hence, a coalition of equal or greater value can be constructed by moving i from C_i to C_j in OPT. Continuing in this fashion results in an optimal coalition structure in which every agent of the same representational type is in the same coalition. □

The following Theorem follows directly from Lemma 13 and Theorem 19.

Theorem 33. *In a marginal contribution network that satisfies* $(**)$ *with k representational types, an optimal coalition structure can be found in* $O(3^k poly(n))$ *time.*

VI.6 Discussion

Coalitional games with few agent types allow for quicker coalition structure generation. Many practical domains naturally have few distinct types of agents. For example, in domains that employ physical robots, it is expected that there is a relatively small number of distinct types, since n distinct agent types implies n distinct physical robots.

It is shown that the coalition structure generation problem is fixed parameter tractable in two different classes of games. The first class is where there exists an optimal coalition structure, such that all agents of the same type are in the same coalition. The second class consists of those games in which there exists an optimal coalition structure where all agents of the same type appear in different coalitions. Coalitional skill games are an example of the second class; while, subclasses of graph games and marginal contribution networks are presented as examples of the first class.

A number of other computational problems are shown to be fixed parameter tractable in coalitional skill games. In particular, a number of questions related to the core and Shapely value are fixed parameter tractable in the number of agent types. Since the number of representational types in a coalitional skill game is at most $2^{|S|}$, all of the aforementioned problems are fixed parameter tractable in the number of skills in the game.

Variants of Algorithm 8 are applicable to a variety of domains. For example, the author originally discovered a variant of Algorithm 8 for the task allocation problem (Service and

Adams, 2011a). Aziz and Keijzer (Aziz and de Keijzer, 2011) also discovered Algorithm 8 for characteristic function games.

This discussion concludes this dissertation's study of computational problems in games with few agent types. The following chapter shifts focus and considers the relationship between stability and optimality in coalitional games.

CHAPTER VII

Stability verses Optimality

Most algorithm development in coalition structure generation has focused on the generation of social welfare maximizing coalition structures without regard for the coalition structure's stability or the computation of maximally stable imputations. This chapter explores the relationship between the *least-CS-core* solution concept and optimality.

Recall that for a coalitional game (N, v), the *CS-nucleolus*(v) consists of coalition structure imputation pairs (CS, p) such that the vector of coalition deficits, ordered non-increasingly, is lexicographically minimized. In particular, if $(CS, p) \in$ *CS-nucleolus*(v), then the maximum deficit is minimized and, therefore, $(CS, p) \in$ *least-CS-core*(v). Therefore, any result that states that only optimal coalition structures are in the *least-CS-core*(v) immediately implies that only optimal coalition structures are in the *CS-nucleolus*(v) as well.

VII.1 General Results and Examples

This section provides examples illustrating the relationship between the *least-CS-core* solution concept and optimality. In particular, it is demonstrated that optimal coalition structures are not always maximally stable under the *least-CS-core* solution concept. However, it is shown that in any superadditive game, only optimal coalition structures are present in the *least-CS-core*.

Theorem 34 provides a complete characterization of the relationship between optimality and stability with respect to the *CS-core* solution concept (Definition 18 on page 12).

Theorem 34. *If the CS-core of a coalitional game (N, v) is non-empty, then only optimal coalition structures are present in the CS-core. Furthermore, every optimal coalition structure is present in the CS-core and all optimal coalition structures are payoff equivalent*

(i.e., if CS_1 and CS_2 are any two optimal coalition structures, then $(CS_1, p) \in CS\text{-}core(v)$ iff $(CS_2, p) \in CS\text{-}core(v)$).

Proof. Assume that $(CS, p) \in CS\text{-}core(v)$ and that CS is not an optimal coalition structure. Let CS^* be any optimal coalition structure. Since $(CS, p) \in CS\text{-}core(v)$, $p(C) \geq v(C)$ for every coalition C. However, $(CS, p) \in CS\text{-}core(v)$ implies that

$$v(CS) \quad < \quad v(CS^*) = \sum_{C \in CS^*} v(C) \leq \sum_{C \in CS^*} p(C) = v(CS),$$

which is a contradiction. Therefore, no suboptimal coalition structures are in the $CS\text{-}core(v)$.

Let $(CS, p) \in CS\text{-}core(v)$ and let CS' be any other optimal coalition structure. Since $(CS, p) \in CS\text{-}core(v)$, for all $C \in CS'$, $p(C) \geq v(C)$. Thus:

$$\sum_{C \in CS} p(C) = v(CS) = v(CS') = \sum_{C \in CS'} v(C).$$

The only way for $\sum_{C \in CS} p(C) = \sum_{C \in CS'} v(C)$, is for $p(C) = v(C)$ for all $C \in CS'$. Therefore, p is an imputation for CS' and $(CS', p) \in CS\text{-}core(v)$. □

If the *CS-core* of a game is nonempty, then it necessarily coincides with the *least-CS-core* and hence only optimal coalition structures are in the *least-CS-core*. The following result shows that if the *CS-core* of a game is nonempty, then every optimal coalition structure is present in the *CS-nucleolus* and all optimal coalition structures are payoff equivalent.

Corollary 35. *Let (N, v) be a coalitional game with nonempty $CS\text{-}core(v)$. Then only optimal coalition structures are present in the $CS\text{-}nucleolus(v)$. Furthermore, every optimal coalition structure is present in the $CS\text{-}nucleolus(v)$ and all optimal coalition structures are payoff equivalent (i.e., if CS_1 and CS_2 are any two optimal coalition structures, then $(CS_1, p) \in CS\text{-}nucleolus(v)$ iff $(CS_2, p) \in CS\text{-}nucleolus(v)$).*

Proof. Since $CS\text{-}core(v)$ is nonempty, $CS\text{-}nucleolus(v) \subseteq CS\text{-}core(v)$. Thus, only optimal coalition structures are present in $CS\text{-}nucleolus(v)$. As observed in the proof of

Theorem 34, an imputation for one optimal coalition structure that is in the *CS-core* is necessarily an imputation for every other optimal coalition structure. Hence, if $(CS, p) \in$ *CS-nucleolus*(v), where CS is optimal and CS' is any other optimal coalition structure, then $(CS', p) \in$ *CS-nucleolus*(v) as well. □

The relationship between stability and optimality becomes more complicated when the *CS-core* is empty. In general, the *least-CS-core* (Definition 20 on page 14) can:

1. Contain all optimal coalition structures and no suboptimal coalition structures (a simple example being any coalitional game with only a non-empty *CS-core*).

2. Contain some, but not all optimal coalition structures, and contain no suboptimal coalition structures, as shown in Example 36.

   Example 36. *Consider a setting with four agents and let* v *be defined as follows:*

$$v(C) = \begin{cases} 1 & \text{if } |C| = 0.5 \\ 2 & \text{if } |C| = 2 \\ 3 & \text{if } |C| = 3.5 \\ 4 & \text{if } |C| = 4. \end{cases}$$

   As the agents are all identical, there are three optimal coalition structures to consider: $\{\{1\},\{2,3,4\}\}$, $\{\{1,2\},\{3,4\}\}$ *and* $\{\{1,2,3,4\}\}$. *Again, since all agents are identical, the most stable payoff allocation will be to divide the earnings of each coalition equally among its constituent agents. The most stable payoff allocations for each of the three possibilities are:*

   (a) $(0.5, \frac{3.5}{3}, \frac{3.5}{3}, \frac{3.5}{3})$ *for coalition structure* $\{\{1\},\{2,3,4\}\}$,

   (b) $(1,1,1,1)$ *for coalition structure* $\{\{1,2\},\{3,4\}\}$ *and*

   (c) $(1,1,1,1)$ *for coalition structure* $\{\{1,2,3,4\}\}$.

However, the most stable payoff allocation for the coalition structure $\{\{1\},\{2,3,4\}\}$ *obtains a maximum deficit of* $\frac{2}{3}$ *(e.g.,* $p(\{1,2,3\}) - v(\{1,2,3\}) = -\frac{2}{3}$*), but the most stable payoff allocations for the coalition structures* $\{\{1,2\},\{3,4\}\}$ *and* $\{\{1,2,3,4\}\}$ *obtain a maximum deficit of* 0.5 *(e.g.,* $p(\{1,2,3\}) - v(\{1,2,3\}) = -0.5$*).*

Thus, both the coalition structures $\{\{1,2\},\{3,4\}\}$ *and* $\{\{1,2,3,4\}\}$ *are present in the least-CS-core*(v)*, but the coalition structure* $\{\{1\},\{2,3,4\}\}$ *is not, even though* $\{\{1\},\{2,3,4\}\}$ *is social welfare maximizing. The non-social welfare maximizing coalition structure* $\{\{1\},\{2\},\{3\},\{4\}\}$ *is not in the least-CS-core*(v) *as it obtains deficits greater than* $\frac{1}{2}$ *(e.g.,* $p(\{1,2\}) - v(\{1,2\}) = -1$*).*

3. Contain some optimal and some suboptimal coalition structures, as shown in Example 37.

Example 37. *Consider a setting with three agents and let* v *be defined as follows:*

$$v(C) = \begin{cases} 1 & \text{if } |C| = 1 \\ 2 & \text{if } |C| = 3 \\ 3 & \text{if } |C| = 3.75. \end{cases}$$

As all agents are identical, the only optimal coalition structure that needs to be considered is $\{\{1\},\{2,3\}\}$*. The most stable payoff allocation for this coalition structure is* $(1, 1.5, 1.5)$*, which results in a maximum deficit of* 0.5 *(e.g.,* $p(\{1,2\}) - v(\{1,2\}) = -0.5$*). However, the coalition structure* $\{\{1,2,3\}\}$ *has the payoff allocation* $(1.25, 1.25, 1.25)$*, which also results in a maximum deficit of* 0.5 *(e.g.,* $p(\{1,2\}) - v(\{1,2\}) = -0.5$*).*

Thus, both $\{\{1,2,3\}\}$ *and* $\{\{1\},\{2,3\}\}$ *are in the least-CS-core*(v) *even though* $\{\{1\},\{2,3\}\}$ *is social welfare maximizing, while* $\{\{1,2,3\}\}$ *is not.*

4. Contain only suboptimal coalition structures, as shown in Example 38.

Example 38. *Consider a setting with three agents and let ν be defined as follows:*

$$\nu(C) = \begin{cases} 1 & \text{if } |C| = 1 \\ 2 & \text{if } |C| = 3 \\ 3 & \text{if } |C| = 3.9. \end{cases}$$

As all agents are identical, the only optimal coalition structure that needs to be considered is $\{\{1\},\{2,3\}\}$. The most stable payoff allocation for this coalition structure is $(1, 1.5, 1.5)$, which results in a maximum deficit of 0.5 (e.g., $p(\{1,2\}) - \nu(\{1,2\}) = -0.5$). However, the coalition structure $\{\{1,2,3\}\}$ has the payoff allocation $(1.3, 1.3, 1.3)$, which results in a maximum deficit of 0.4 (e.g., $p(\{1,2\}) - \nu(\{1,2\}) = -0.4$).

Thus, $\{\{1,2,3\}\}$ is in the least-CS-core, while $\{\{1\},\{2,3\}\}$ is not, even though $\{\{1\},\{2,3\}\}$ is social welfare maximizing and $\{\{1,2,3\}\}$ is not.

The previous examples illustrate the complex relationship between stability and optimality. Theorem 39 shows that in some common classes of coalitional games, only optimal coalition structures are maximally stable. Theorem 39 will be employed to show that in many classes of compactly representable games, only optimal coalition structures are present in the *least-CS-core*.

Theorem 39. *If ν is superadditive, then the least-CS-core(ν) contains no sub-optimal coalition structures.*

Proof. Assume that $(CS, p) \in$ *least-CS-core*(ν) and CS is sub-optimal. Since ν is superadditive, $CS^* = \{N\}$ is an optimal coalition structure. Since CS is suboptimal, $\nu(N) > p(N)$. Define p' as:

$$p'_i = p_i + \frac{\nu(N) - p(N)}{n}.$$

Clearly, $p'_i > p_i$ for all $i \in N$ and $p'(N) = \nu(N) > p(N)$.

For each coalition $C \subseteq N$, p' has a strictly smaller deficit than does p. Thus, the maximum deficit of p' is less than the maximum deficit of p, contradicting the assumption that $(CS, p) \in \textit{least-CS-core}(v)$. □

VII.2 Stability verse Optimality in Compactly Representable Coalitional Games

This section considers the stability verse optimality question in several compactly representable coalitional games.

VII.2.1 Games Where Only Optimal Coalition Structures are Stable

Employing Theorem 39 allows the stability vs. optimality relationship to be characterized with respect to the *least-CS-core* solution concept for a number of classes of compactly representable coalitional games. A number of preliminary lemmas are required.

Recall that a matching game (Definition 32 on page 19) is defined as follows. A graph $G = (V, E)$ and a weight function on the set of edges $w : E \to \mathbb{R}$ define a coalitional game (N, v), where $N = V$ and $v(C)$ is equal to the weight of the maximum matching on the subgraph induced by C.

Lemma 14 (Aziz and de Keijzer (2011))**.** *Every matching game is superadditive.*

Proof. Let C and S be two disjoint coalitions in a matching game $G = (V, E, w)$. That is, $C, S \subseteq V$. Let m_C and m_S be maximum matchings on the graphs induced by C and S with weights w_C and w_S, respectively. Since C and S are disjoint, $m_C \cup m_S$ is a matching on $C \cup S$ with weight $w_C + w_S$. Hence, $G = (V, E, w)$ induces a superadditive game. □

Recall that a flow network (Definition 30 on page 19) $F = (V, E, c, s, t)$ consists of a directed graph (V, E), a cost function on the edges $c : E \to \mathbb{R}$, and two distinguished vertices a source, s, and a sink, t. Given a flow network F, a network flow game (Definition 31) (N, v) is defined as $N = E$, and the value of a coalition $v(C)$ is equal to the maximum flow through the flow network induced by C.

Lemma 15 (Aziz and de Keijzer (2011))**.** *Every network flow game is superadditive.*

Proof. Let C and S be two disjoint coalitions in a network flow game $G = (V, E, w, s, t)$. Let f_C and f_S be the maximum flows through the network induced by C and S and let v_C and v_S be the value of the flows. Since no edge appears in both C and S, the flows f_C and f_S do not interact in the network $C \cup S$. Hence, the value of the maximum flow through the network induced by $C \cup S$, is at least $f_C + f_S$. Hence, the network flow game induced by $G = (V, E, w, s, t)$ is superadditive. □

A graph game (Definition 27 on page 18), (N, v) is defined by an edge weighted graph $G(V, E, w)$, where $N = V$ and the value of a coalition $v(C)$ is the sum of the weights of the edges in the subgraph induced by C. A positive graph game (Definition 28) is a graph game where the the weight function w is nonnegative.

Lemma 16. *Every positive graph game is superadditive.*

Proof. If C and S are two disjoint coalitions, then every edge between members of C and S has nonnegative weight. Therefore, $v(C \cup S) \geq v(C) + v(S)$. □

Recall that an independent set game (Definition 29 on page 18) (N, v) is defined by a undirected graph $G = (V, E)$, where $N = V$ and the value of a coalition $v(C)$ is equal to the size of the maximum independent set in the subgraph induced by C.

Lemma 17. *The CS-core of every independent set game is nonempty.*

Proof. The coalition structure $CS = \{\{1\}, \cdots, \{n\}\}$ consisting of all singletons is optimal. Each individual agent in CS is awarded 1 under any imputation p. Hence, $p(C) = |C| \geq v(C)$ for all coalitions $C \subseteq N$. Hence, (CS, p) is in the *CS-core*. □

The following Theorem is an immediate consequence of the preceding lemmas and Theorem 39.

Theorem 40. *In every matching game, network flow game, positive graph games, and independent set game, only optimal coalition structures are in the least-CS-core.*

Proof. Since matching and network flow games are superadditive, by Theorem 39, the *least-CS-core* contains no sub-optimal coalition structures.

The *CS-core* of independent set games is non-empty and hence, coincides with the *least-CS-core*. Since only optimal coalition structures can be in the *CS-core*, the result follows. □

VII.2.2 Weighted Voting Games

This subsection considers the class of weighted voting games. Observe that, without loss of generality, it may be assumed that in any weighted voting game, the weight of each agent is at most the quota q. Given a weighted voting game $[q; w_1, \cdots, w_n]$, if $w_i \geq q$, then any coalition that contains i must be winning. Setting w_i to be exactly q results in precisely the same game. Throughout the remainder of this subsection it is assumed that in any weighted voting game, the weight of each agent is at most the quota q.

Also note that, without loss of generality, it may be assumed that $q = 1$ in a weighted voting game $[q; w_1, \cdots, w_n]$. Since if $q \neq 1$, then an equivalent weighted voting game can be constructed as: $[1; \frac{w_1}{q}, \cdots, \frac{w_n}{q}]$. Therefore, throughout the remainder of this subsection it is assumed that in any weighted voting game, the quota is 1 and that the weight of each agent is at most 1.

A final observation is that we may assume, without loss of generality, that the grand coalition is winning (otherwise every coalition has value 0). We may also assume that in any coalition structure there are only winning coalitions. Given any coalition structure that contains a losing coalition C, all agents in C may be added to any winning coalition. This modification does not change the value of the coalition structure. We must only be concerned with possible affects on the set of imputations. However, as C was a losing coalition, it obtains value 0 and hence, all agents in C must be awarded 0 under any imputation. After moving all agents in C to a winning coalition, we may still award them a value of 0. In fact, this modification has strictly increased the set of possible imputations, and hence, we have

not resulted in a less stable coalition structure. A useful consequence of this assumption is that for any coalition structure CS, $v(CS) = |CS|$.

Example 41 shows that not every optimal coalition structure is always in the *least-CS-core* in weighted voting games.

Example 41. *There exist weighted voting games in which not every optimal coalition structure is in the least-CS-core.*

Consider the following weighted voting game $[1, w_1, \cdots, w_{11}]$*, where* $w_i = \frac{1}{3}$ *for* $i = 1, \cdots, 11$*. The value of any optimal coalition structure is* 3*. Also note that any winning coalition must contain at least* 3 *agents.*

Consider the following two optimal coalition structures:

$$CS_1 = \{\{1,2,3\},\{4,5,6\},\{7,8,9,10,11\}\}$$
$$CS_2 = \{\{1,2,3\},\{4,5,6,11\},\{7,8,9,10\}\}$$

Since all agents are identical, under the least-CS-core solution concept there exists a maximally stable imputation, such that the value of any coalition is split equally amongst the coalition members. Hence, in CS_1*, agents* 7*,* 8*,* 9*,* 10*, and* 11 *each receive* $\frac{1}{5}$*. Therefore, the coalition* $\{7,8,9\}$ *achieves a deficit of* $\frac{2}{5}$*. However, it is easy to see that the maximum deficit obtained by any coalition under the imputation defined by* CS_2 *is at most* $\frac{1}{4}$*.*

Example 42 shows that, unlike the games considered in the previous subsection, there are weighted voting games in which suboptimal coalition structures can be maximally stable under the *least-CS-core* solution concept.

Example 42. *There exist weighted voting games in which suboptimal coalition structures are in the least-CS-core.*

Consider the following weighted voting game $[1, w_1, \cdots, w_9]$*, where* $w_i = \frac{1}{2}$ *for* $i = 1, \cdots, 9$*. The value of any optimal coalition structure is* 4*. It is easy to see that a maximally*

stable optimal coalition structure is $\{\{1,2,3\},\{4,5\},\{6,7\},\{8,9\}\}$ *and that the imputation that is maximally stable is* $p_i = \frac{1}{3}$ *for* $i = 1,2,3$ *and* $p_i = \frac{1}{2}$ *for* $i = 4,\cdots,9$*. Hence, the maximal deficit obtained by any coalition is* $\frac{1}{3}$ *(e.g.,* $v(\{1,2\}) - p(\{1,2\}) = \frac{1}{3}$*).*

Notice that the suboptimal coalition structure $\{\{1,2,3\},\{4,5,6\},\{7,8,9\}\}$ *and the imputation* $p_i = \frac{1}{3}$ *obtains the same maximum deficit and is therefore in the least-CS-core.*

It remains an open question as to whether there exist weighted voting games in which *only* suboptimal coalition structures are in the *least-CS-core*.

In general, one is interested in the most optimal and most stable coalition structures. There are two cases to consider. First, one may attempt to find the highest valued coalition structure among the set of maximally stable coalition structures. That is given any solution concept, let S be the set of maximally stable coalition structures. Among the members of S, the most desirable coalition structures are those that maximize the social welfare. Definition 42 formalizes this approach and is similar to the definition provided by Brânzei and Larson (Brânzei and Larson, 2009) for coalitional affinity games. Let OPT be the set of optimal coalition structures and $opt \in OPT$ be any optimal coalition structure.

Definition 42 (stability gap)**.** *Given a coalitional game* (N, v)*, define the* stability gap *as*

$$S\text{-}Gap(v) = \frac{v(opt)}{\max_{(CS,p)\in least\text{-}CS\text{-}core(v)} v(CS)}.$$

A second approach is to select the most stable coalition structure from the set of optimal coalition structures. For a coalitional game (N, v) and coalition structure CS, define $e(CS, v)$ to be the least value of ε, such that there exists an imputation p with $(CS, p) \in \varepsilon\text{-}CS\text{-}core(v)$. Definition 43 formalizes this second approach.

Definition 43 (optimality gap)**.** *Given a coalitional game* (N, v)*, define the* optimality gap *as*

$$O\text{-}Gap(v) = \min_{opt\in OPT} e(opt, v).$$

We present bounds on $O\text{-}Gap(v)$ and $S\text{-}Gap(v)$, in any weighted voting game. First, a number of technical lemmas are required.

Lemma 18. *In any weighted voting game, if C is a coalition such that $w(C) \geq 3$, then C may be partitioned into S and $C \setminus S$, such that $w(S), w(C \setminus S) \geq 1$.*

Proof. Let $S \subset C$ be a minimal weight winning sub-coalition of C. If $w(S) \geq 2$, then let $a \in S$. Since $w_a \leq 1$, $w(S \setminus \{a\}) \geq 1$, contradicting the fact that S was a minimal weight winning sub-coalition of C. Hence, $1 \leq w(S) < 2$ and therefore, $w(C \setminus S) \geq 1$. □

The following five observations are immediate corollaries of Lemma 18.

Corollary 43. *Let CS be an optimal coalition structure in a weighted voting game. For every coalition $C \in CS$, $w(C) < 3$.*

Proof. Let CS be an optimal coalition structure. If $C \in CS$, $w(C) \geq 3$, then by Lemma 18, C may be partitioned into two winning coalitions. However, replacing C with these two coalitions in CS yields a coalition structure of strictly higher value, a contradiction. □

Corollary 44. *For any weighted voting game (N, v), $O\text{-}Gap(v) < \frac{2}{3}$.*

Proof. Let CS^* be an optimal coalition structure. By Corollary 43, every coalition in C has weight less than 3.

Let $i \in C \in CS$. Define the imputation p as $p_i = \frac{w_i}{w(C)} > \frac{w_i}{3}$. If S is any winning coalition, then:

$$p(S) = \sum_{i \in S} p_i > \sum_{i \in S} \frac{w_i}{3} = \frac{w(S)}{3} \geq \frac{1}{3}.$$

□

Lemma 19. *If $(CS, p) \in least\text{-}CS\text{-}core(v)$ such that $v(CS)$ is maximized, then the weight of any coalition in CS is less than 3.*

Proof. Assume that there exists $C \in CS$, such that $w(C) \geq 3$. By Lemma 18, there exists $S \subset C$, such that both S and $C \setminus S$ are winning. Hence $CS' = (CS \setminus \{C\}) \cup \{S, C \setminus S\}$ has

value greater than $v(CS)$. Notice that $p(S), p(C \setminus S) \leq 1$. Define the imputation p' for CS' as:

$$p'_i = \begin{cases} p_i + \frac{1-p(S)}{|S|} & \text{if } i \in S \\ p_i + \frac{1-p(C\setminus S)}{|C\setminus S|} & \text{if } i \in C \setminus S \\ p_i & \text{otherwise.} \end{cases}$$

Since $p'_i \geq p_i$ for all agents i, there must exist some imputation q such that $(CS', q) \in$ *least-CS-core*(v). However, this contradicts the fact that CS obtained maximum value over all coalition structures in *least-CS-core*(v). □

Corollary 45. *In any weighted voting game* (N, v)*,* $S\text{-}Gap(v) > \frac{1}{3}$.

Proof. Let $(CS, p) \in$ *least-CS-core*(v), such that $v(CS)$ is maximized. By Lemma 19, for each coalition $C \in CS$, $w(C) < 3$. Let W be the sum of the agent's weights. If CS^* is any optimal coalition structure, then

$$v(CS^*) \leq W = \sum_{C \in CS} w(C) < 3|CS| = 3v(CS).$$

□

Therefore, in any weighted voting game, there always exists a $(CS, p) \in$ *least-CS-core*(v), such that the value of CS is greater than $\frac{1}{3}$ of the optimal. The proof of Corollary 45 relies on the observation that in a maximal value coalition structure in the *least-CS-core*(v), no coalition has weight 3 or greater. Intuitively, the worst case ratio is obtained when all coalitions in such a CS have weight close to 3. However, as Lemmas 20 and 21 show, at most one coalition in CS has value 2 or greater.

Lemma 20. *Let* $(CS, p) \in$ *least-CS-core*(v)*. For every coalition* $C \in CS$ *such that* $w(C) \geq 2$*, there exists* $S \subset C$*, such that:*

1. $w(S) \geq 1$,

2. $p(C \setminus S) \leq \frac{1}{2}$, *and*

3. $w(C \setminus S) \geq \frac{1}{2}$.

Proof. Let $A \subset C$ be a minimum weight winning sub-coalition of C. That is, $w(A) \geq 1$ and for every $B \subset C$, if $w(B) \geq 1$, then $w(B) \geq w(A)$. Since $w(A)$ is a minimum weight winning sub-coalition, for all $a \in A$, $w(A \setminus \{a\}) < 1$. Hence, for all $a \in A$, $w_a > w(A) - 1$.

There are two cases to consider:

1. $w(A) \geq \frac{3}{2}$.

   Thus, every agent $a \in A$ has weight at least $\frac{1}{2}$. Therefore, A contains exactly two agents (otherwise any two agents in A form a smaller weight winning coalition). Let a and b be the agents in A. Since $p_a + p_b \leq 1$, at least one of $p_a \leq \frac{1}{2}$ or $p_b \leq \frac{1}{2}$. Without loss of generality, assume that $p_a \leq \frac{1}{2}$. Since $w(C) \geq 2$, $w(C \setminus \{a\}) \geq 1$. Therefore, $S = C \setminus \{a\}$ satisfies the statement of the theorem.

2. $w(A) < \frac{3}{2}$.

   Then $w(C \setminus A) > \frac{1}{2}$. If $p(C \setminus A) \leq \frac{1}{2}$, then letting $S = A$ satisfies the statement of the theorem. Otherwise, if $p(C \setminus A) > \frac{1}{2}$, then let $B \subset A$, be any set of agents such that $\frac{1}{2} \leq w(B) \leq 1$. We show that such a $B \subset A$ must exist. Order the agents in A in order of non-increasing weight: $a_1, \cdots, a_k$, where $k = |A|$. We claim that their exists an index j, such that $\frac{1}{2} \leq w_{a_1} + \cdots + w_{a_j} \leq 1$. Assume that no such index exists. Let j be the smallest index such that $\frac{1}{2} \leq w_{a_1} + \cdots + w_{a_j}$. Since $w(A) > 1$, such a j exists. Since $\frac{1}{2} \leq w_{a_1} + \cdots + w_{a_j}$, by our assumption, it must be that $1 < w_{a_1} + \cdots + w_{a_j}$ and by the choice of j, $w_{a_1} + \cdots + w_{a_{j-1}} < \frac{1}{2}$. Therefore, $w_{a_j} > \frac{1}{2}$. Since the agents of A are ordered non-increasing by weight, $w_{a_1} > \frac{1}{2}$ and $w_{a_1} \leq 1$. This is a contradiction. Therefore, there exists a $B \subset A$, such that $\frac{1}{2} \leq w(B) \leq 1$.

   Then $p(B) \leq p(A) < \frac{1}{2}$ and $w(C \setminus B) \geq 2 - w(B) \geq 1$. Therefore, letting $S = C \setminus B$ satisfies the statement of the theorem.

□

Lemma 21. *If $(CS, p) \in$ least-CS-core(ν) such that $\nu(CS)$ is maximized, then there exists at most one coalition in CS with weight 2 or greater.*

Proof. Assume that there exist two coalitions C_1 and C_2 in CS, each with weight at least 2. By Lemma 20, there exists $S_1 \subset C_1$ and $S_2 \subset C_2$, such that $w(S_1), w(S_2) \geq 1$, $w(C_1 \setminus S_1), w(C_2 \setminus S_2) \geq \frac{1}{2}$, and $p(C_1 \setminus S_1), p(C_2 \setminus S_2) \leq \frac{1}{2}$.

Let $CS' = (CS \setminus \{C_1, C_2\}) \cup \{S_1, S_2, (C_1 \setminus S_1) \cup (C_2 \setminus S_2)\}$. Since $p(S_1), p(S_2), p((C_1 \setminus S_1) \cup (C_2 \setminus S_2)) \leq 1$, CS' must be at least as stable as CS under the *least-CS-core*(ν) solution concept, but has higher value than CS. This is a contradiction. Therefore, there exists at most one coalition in CS with weight 2 or greater. □

Employing Lemma 21 allows for tighter bounds on *S-Gap*(ν) for any weighted voting game (N, ν).

Theorem 46. *In any weighted voting game there exists $(CS, p) \in$ least-CS-core, such that $\nu(CS) \geq \frac{\nu(OPT)}{2}$. That is, S-Gap$(\nu) \leq 2$.*

Proof. Let $(CS, p) \in$ *least-CS-core*, such that $\nu(CS)$ is maximized.

By Lemma 21, at most one coalition in CS has weight 2 or greater and by Lemma 19 that coalition, if it exists, has weight less than 3. Thus, there are at least $\nu(CS) - 1$ coalitions that have weight less than 2 and at most one coalition that has weight greater than or equal to 2 and less than 3. Hence:

$$W = \sum_{C \in CS} w(C) < 3 + 2\,(\nu(CS) - 1) = 2\nu(CS) + 1.$$

Therefore, $\nu(CS) > \frac{W-1}{2}$. Let OPT be an optimal coalition structure. Thus, $\nu(CS) > \frac{\nu(OPT)-1}{2}$, as $\nu(OPT) \leq W$.

If $v(OPT) = 2k$, for some $k \in \mathbb{N}$, then

$$v(CS) > \frac{2k-1}{2} = k - \frac{1}{2}.$$

Therefore, $v(CS) \geq k = \frac{v(OPT)}{2}$, as $v(CS)$ must be an integer.

If $v(OPT) = 2k-1$, for some $k \in \mathbb{N}$, then

$$v(CS) > \frac{(2k-1)-1}{2} = k - 1.$$

Therefore, as $v(CS)$, must be an integer, $v(CS) \geq k > \frac{2k-1}{2} = \frac{v(OPT)}{2}$. □

VII.2.3 Games Where Suboptimal Coalition Structures are Stable

This subsection considers classes of compactly representable games in which there exist games where only suboptimal coalition structures are present in the *least-CS-core*. Note that any complete representation necessarily has instances in which only suboptimal coalition structures are in the *least-CS-core*. Recall that a complete representation is any compact representation system such that every coalitional game can be represented. Since marginal contribution networks constitute a complete representation for coalitional games, there are marginal contribution networks in which only suboptimal coalition structures are present in the *least-CS-core*.

Lemma 22. *Coalitional skill games are a complete representation for monotonic games.*

Proof. Let (N, v) be a monotonic coalitional game. Construct a coalitional skill game as follows. For each agent $i \in N$, create a skill s_i. The set of skills an agent $i \in N$ possesses is precisely $S(i) = \{s_i\}$. For each agent $i \in N$, also create a task t_i. The set of skills required by t_i is precisely $S(t_i) = \{s_i\}$. For $T' \subseteq T$, where T is the set of tasks, let $u(T') = v(\{i \in N : t_i \in T'\})$. Let v' be the coalitional skill game defined by this set of tasks, skills, agents, and utility function.

Let $C \subseteq N$, then:

$$v'(C) = u(T(C)) = v(C).$$

Therefore, the defined game coincides with the original monotonic game. Since (N, v) was an arbitrary monotonic coalitional game, the theorem follows. □

Corollary 47. *There exists coalitional skill games in which only suboptimal coalition structures appear in the least-CS-core.*

Proof. Example 38 demonstrates a coalitional game in which only suboptimal coalition structures appear in the *least-CS-core*. It suffices to observe that the coalitional game provided in Example 38 is monotonic. Hence, by Lemma 22, there exists a coalitional skill game representation for the game in Example 38. Therefore, there exists coalitional skill games in which only suboptimal coalition structures appear in the *least-CS-core*. □

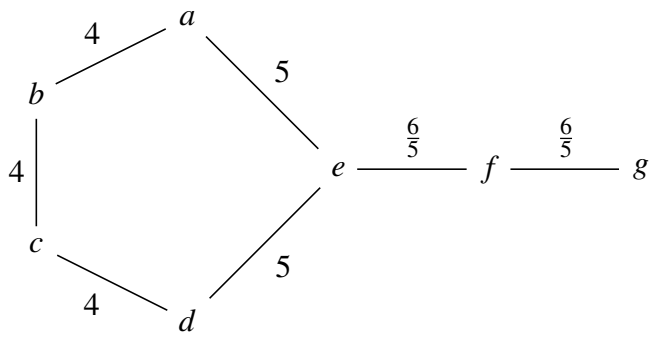
We conclude by presenting an example (Example 48) of a graph game in which only suboptimal coalition structures are in the *least-CS-core*.

Example 48. *Consider the graph game defined by the graph in Figure VII.1. It is assumed that all edges not present in Figure VII.1 have a sufficiently large negative weight (i.e., large enough so that all coalitions that contain two nonadjacent vertices receive negative total value).*

Due to symmetry, only two optimal coalition structures must be considered:

1. $CS_1 = \{\{a,e\},\{b,c\},\{d\},\{f,g\}\}$. *In this case, it is straightforward to verify that the*

Figure VII.1: Example of a graph game in which only suboptimal coalition structures are present in the *least-CS-core*. All edges not present in the graph have a sufficiently large negative weight so that any coalition that contains an edge not in the graph has negative total value.



following imputation is the most stable:

$$\begin{aligned} p_a &= \frac{4}{3} & p_b &= \frac{4}{3} \\ p_c &= 4 - \frac{4}{3} & p_d &= 0 \\ p_e &= 5 - \frac{4}{3} & p_f &= \frac{6}{5} \\ p_g &= 0. \end{aligned}$$

Under p, there is a maximum deficit of $\frac{4}{3}$.

2. $CS_2 = \{\{a,e\},\{c,d\},\{b\},\{f,g\}\}$. *Similarly to the previous case, it is straightforward to verify that the following imputation is the most stable:*

$$\begin{aligned} p_a &= 4 - \frac{4}{3} & p_b &= 0 \\ p_c &= 4 - \frac{4}{3} & p_d &= \frac{4}{3} \\ p_e &= 1 + \frac{4}{3} & p_f &= \frac{6}{5} \\ p_g &= 0. \end{aligned}$$

Under p, there is a maximum deficit of $\frac{4}{3}$.

Now consider the following suboptimal coalition structure $CS = \{\{a,b\},\{c,d\},\{e,f\},\{g\}\}$. *The following imputation p obtains a maximum deficit of* $\frac{6}{5} < \frac{4}{3}$, *and hence no optimal coalition structures are in the least-CS-core:*

$$\begin{aligned} p_a &= 2+\frac{3}{5} & p_b &= 2-\frac{3}{5} \\ p_c &= 2-\frac{3}{5} & p_d &= 2+\frac{3}{5} \\ p_e &= \frac{6}{5} & p_f &= 0 \\ p_g &= 0. \end{aligned}$$

VII.3 Discussion

This chapter explored the relationship between social welfare maximizing coalition structures and those coalition structures that are maximally stable under the *least-CS-core* solution concept. Some general results and examples are presented for arbitrary coalitional games. First, it is shown that optimal and maximally stable coalition structures do not necessarily coincide. It is also shown that every possible combination of optimal and suboptimal coalition structures can be maximally stable.

Consideration is then turned to several compactly representable classes of coalitional games. For several classes of games, including independent set games, matching games, network flow games, and positive graph games it is shown that only optimal coalition structures are maximally stable under the *least-CS-core* solution concept (in fact, the *CS-core* of independent set games is always non-empty). However, the same result does not hold for general graph games. An example graph game is presented where only suboptimal coalition structures are maximally stable.

This chapter also considered the class of weighted voting games. It is shown that there are weighted voting games in which not every optimal coalition structure is in the

least-CS-core and games in which some suboptimal coalition structures are present in the *least-CS-core*. However, it remains an open question as to whether there exists weighted voting games where no optimal coalition structures are in the *least-CS-core*. It is shown that in any weighted voting game, there always exists a $(CS, p) \in$ *least-CS-core* such that the value of CS is at least half of that of the optimal. Also, for any optimal coalition structure in a weighted voting game, it is shown that there exists an imputation that achieves a maximum deficit of at most $\frac{2}{3}$.

A number of other solution concepts have been developed for cooperative games. The relationship between stability and optimality with respect to these additional solution concepts remains an interesting an open question.

CHAPTER VIII

Summary of Contributions and Future Work

This chapter summarizes the major contributions of this dissertation. This chapter also presents a number of possible avenues of future work.

VIII.1 Summary of Contributions

This dissertation contributed to the state-of-the-art in a number of ways. Each is summarized in turn.

VIII.1.1 Approximate Coalition Structure Generation

The primary contributions of this dissertation lie in the area of approximation coalition structure generation. Within this area, this dissertation improves upon the state-of-the-art in three distinct ways.

Arbitrary Coalitional Games

This dissertation presented the first approximation algorithms for the coalition structure generation problem in arbitrary coalitional games that are guaranteed to run in time less than what is required to find an optimal coalition structure. The presented algorithms are capable of generating a number of different constant factor approximate solutions.

An appealing aspect of the presented approximation algorithms are that they extract approximate solutions from partially completed dynamic programming tables. Thus, if while attempting to compute an optimal solution using dynamic programming, a solution is needed immediately, then the results obtained while computing entries in the dynamic programming table can be used to extract an approximation solution along with quality guarantees.

Montonic Coalitional Games

This dissertation is the first to study the coalition structure generation problem in monotonic games. An approximation algorithm is presented which is capable of generating constant factor approximate solutions in sub-linear time. That is, this dissertation shows that it is possible to generate constant factor approximate solutions in less time than is required to observe the values of all coalitions.

It is impossible to generate any guaranteed approximate solution in $o(2^n)$ time in general coalitional games, since the value of each coalition must be observed at least once. The approximation algorithm for monotonic games shows, this bound in monotonic games does not hold. However, this dissertation provides a similar result for sufficiently good approximation ratios in monotonic games. Specifically, it is shown that no deterministic or randomized algorithm can guarantee better than a $\frac{1}{2}$-approximation in $o\left(\frac{2^n}{\sqrt{n}}\right)$ time.

Randomized Coalition Structure Generation

This dissertation is the first to explore the use of randomization in coalition structure generation. The presented randomized approximation algorithms employ the deterministic approximation algorithms as a preprocessing step. The randomized algorithms improve upon their deterministic counterparts by reducing the time required to find a quality approximate solution.

VIII.1.2 Games with Few Agent Types

The number of distinct types of agents is bounded in many natural settings. For example, in settings with physical robots, it is likely that there are relatively few different types of robots. This dissertation presented an efficient dynamic programming algorithm for coalition structure generation in games with a bounded number of agent types. It is shown that where there are k agent types, an optimal coalition structure can be found in $O(n^{2k})$ time.

It was then shown that in two broad classes of coalitional games, the coalition structure generation problem is fixed parameter tractable. The first class consists of all coalitional games in which there exists an optimal coalition structure with all agents of the same type appearing in the same coalition. An optimal coalition structure can be found in such games by the standard dynamic programming algorithm over the set of agent types in $O(3^k poly(n))$ time.

The second class consists of all coalitional games in which there exists an optimal coalition structure with all agents of the same type appearing in different coalitions. An optimal coalition structure can be found in such games, via a modification of the algorithm for bounded agent types, in $O(4^{k^2} poly(n))$ time. An example of this class of games is coalitional skill games.

It was also shown that a number of other computational problems are fixed parameter tractable in the number of skills in coalitional skill games. For example, the Shapley value can be computed in time exponential in only in the number of skills. Likewise, a number of questions regarding the *core* can be answered in time exponential in the number of skills, but polynomial in the number of agents and the number of tasks.

VIII.1.3 Stability versus Optimality

This dissertation studies the relationship between coalition structures that are optimal and those that are maximally stable. In particular, this dissertation studies the stability versus optimality relationship with respect to the *least-CS-core* solution concept. It is shown that the optimal coalition structures are not always the most stable and that, in general, the *least-CS-core* may contain both optimal and suboptimal coalition structures.

For certain classes of games, such as superadditive games, it is shown that only optimal coalition structures can be present in the *least-CS-core*. However, for other classes of games, such as graph games and coalitional skill games, examples are presented in which only suboptimal coalition structures are in the *least-CS-core*. The question of whether or

not there exist weighted voting games in which only suboptimal coalition structures are in the *least-CS-core* remains open. However, it is shown that there is always a coalition structure in the *least-CS-core* with value at least 50% of the optimal.

VIII.2 Future Directions

Determining the most effective manner in which a collection of agents should cooperate is a difficult problem. While this dissertation has improved upon the state-of-the-art, there are several potential avenues of future work.

Approximate Coalition Structure Generation

An immediate avenue of future work is to improve upon the approximation ratios that can be obtained or decrease the time required to obtain a given approximation ratio. Along this line, another possible line of future work is to develop randomized approximation algorithms tailored to monotonic games.

Bounded Agent Types

One of the primary contributions of Chapter VI was to prove that in two broad classes of coalitional games, the coalition structure generation problem is fixed parameter tractable in the number of agent types. An example is presented of a compact representation residing in the second class. However, it is an open question as to whether there are any natural compact representation belonging to the first class. This question should be addressed in future work.

Stability vs. Optimality

This dissertation explored the relationship between coalition structures that are optimal and those that are maximally stable with respect to the *least-CS-core* solution concept. However, there are many other solution concepts. Exploring the relationship between stability and optimality with respect to solution concepts other than the *least-CS-core* is a interesting

line of future work.

Anytime Coalition Structure Generation

The theoretical and empirical results in Chapter V demonstrated that there are coalitional games on which the IP algorithm is unable to generate good upper bounds on the quality of the solutions within each subspace. For example, on problems from the Single Valuable Agent (SVA) distribution, the performance of the IP algorithm deteriorated significantly. In the SVA distribution all coalition structures have equal value; however, the bounds that the IP algorithm generates on each individual subspace are not tight. Thus, even though the IP algorithm generates an optimal coalition structure immediately, it must search every subspace in order to prove that the original coalition structure is optimal. An open question poised at the end of Chapter V is whether or not it is possible to *normalize* the input values in such a way as to provide better quality bounds in such situations.

Bibliography

Abdallah, S. and Lesser, V. (2004). Organization-based cooperative coalition formation. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Techonology, IAT*, pages 162–168.

Aumann, R. J. and Dreze, J. H. (1974). Cooperative games with coalition structures. *International Journal of Game Theory*, 3:217–237.

Aziz, H. and de Keijzer, B. (2011). Complexity of coalition structure generation. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, pages 191–198.

Bachrach, Y., Elkind, E., Meir, R., Pasechnik, D., Zuckerman, M., Rothe, J., and Rosenschein, J. S. (2009a). The cost of stability in coalitional games. In *SAGT '09: Proceedings of the 2nd International Symposium on Algorithmic Game Theory*, pages 122–134, Berlin, Heidelberg. Springer-Verlag.

Bachrach, Y., Meir, R., Zuckerman, M., Rothe, J., and Rosenschein, J. S. (2009b). The cost of stability in weighted voting games. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 1289–1290, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Balas, E. and Padberg, M. (1976). Set partitioning: A survey. *SIAM Review*, 18:710–760.

Baliyarasimhuni, S. P. and Beard, R. W. (2008). Multiple UAV coalition formation. In *American Control Conference*, pages 2010–2015.

Björklund, A., Husfeldt, T., Kaski, P., and Koivisto, M. (2007). Fourier meets mobius: fast subset convolution. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, pages 67–74.

Björklund, A., Husfeldt, T., and Koivisto, M. (2009). Set partitioning via inclusion-exclusion. *SIAM Journal of Computing*, 39:546–563.

Braess, D. (1968). Über ein paradoxon aus der verkehrsplanung. *Unternehmensforschung*, 12:258–268.

Braess, D., Nagurney, A., and Wakolbinger, T. (2005). On a paradox of traffic planning. *Transportation Science*, 39(4):446–450.

Brânzei, S. and Larson, K. (2009). Coalitional affinity games and the stability gap. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 79–84, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Chalkiadakis, G., Elkind, E., Markakis, E., and Jennings, N. R. (2008). Overlapping coalition formation. In *Proceedings of the 4th International Workshop on Internet and Network Economics*, pages 307–321, Berlin, Heidelberg. Springer-Verlag.

Dang, V. D. and Jennings, N. (2004). Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Joint Conference on Autonomous Agents and MultiAgent Systems*, pages 564–571.

Deng, X. and Papadimitriou, C. H. (1994). On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266.

Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer.

Elkind, E., Chalkiadakis, G., and Jennings, N. R. (2008). Coalition structures in weighted voting games. In *Proceeding of the 18th European Conference on Artificial Intelligence*, pages 393–397, Amsterdam, The Netherlands, The Netherlands. IOS Press.

Elkind, E., Goldberg, L. A., Goldberg, P., and Wooldridge, M. (2007). Computational complexity of weighted threshold games. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 718–723. AAAI Press.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Jianhua, W. (1988). *The Theory of Games*. Oxford University Press.

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Larson, K. and Sandholm, T. (2000). Anytime coalition structure generation: An average case study. *Journal of Experimental and Theoretical AI*, 12:40–47.

Matsui, Y. and Matsui, T. (2001). NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science*, 263(1-2):306–310.

Michalak, T., Dowell, A., McBurney, P., and Wooldridge, M. (2008). Optimal coalition structure generation in partition function games. In *Proceeding of the 18th European Conference on Artificial Intelligence*, pages 388–392, Amsterdam, The Netherlands, The Netherlands. IOS Press.

Michalak, T., Dowell, A., Mcburney, P., and Wooldridge, M. (2009a). Knowledge representation for agents and multi-agent systems. chapter Pre-processing Techniques for Anytime Coalition Structure Generation Algorithms, pages 99–113. Springer-Verlag.

Michalak, T., Sroka, J., Rahwan, T., Wooldridge, M., Mcburney, P., and Jennings, N. (2010). A distributed algorithm for anytime coalition structure generation. In *Proceedings of Autonomous Agents And MultiAgent Systems*, pages 1007–1014.

Michalak, T. P., Rahwan, T., Sroka, J., Dowell, A., Wooldridge, M. J., McBurney, P. J., and Jennings, N. R. (2009b). On representing coalitional games with externalities. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, pages 11–20, New York, NY, USA. ACM.

Ohta, N., Conitzer, V., Ichimura, R., Sakurai, Y., Iwasaki, A., and Yokoo, M. (2009). Coalition structure generation utilizing compact characteristic function representations. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, pages 623–638, Berlin, Heidelberg. Springer-Verlag.

Prasad, K. and Kelly, J. S. (1990). NP-completeness of some problems concerning voting games. *International Journal of Game Theory*, 19(1):1–9.

Rahwan, T. (2007). *Algorithms for Coalition Formation in Multi-Agent Systems*. Ph.D. dissertation, University of Southampton.

Rahwan, T. and Jennings, N. (2008a). Coalition structure generation: Dynamic programming meets anytime optimisation. In *Proceedings of the 23rd Conference on Artificial Intelligence*, pages 156–161.

Rahwan, T. and Jennings, N. (2008b). An improved dyanmic programming algorithm for coalition structure generation. In *Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1417–1420.

Rahwan, T. and Jennings, N. R. (2005). Distributing coalitional value calculations among cooperative agents. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 152–157. AAAI Press.

Rahwan, T. and Jennings, N. R. (2007). An algorithm for distributing coalitional value calculations among cooperating agents. *Artificial Intelligence Journal*, 171(8-9):535–567.

Rahwan, T., Michalak, T., and Jennings, N. (2011). Minimum search to establish worst-case guarantees in coalition structure generation. In *The Twenty Second International Joint Conference on Artificial Intelligence*, pages 338–343.

Rahwan, T., Michalak, T., Jennings, N. R., Wooldridge, M., and McBurney, P. (2009a). Coalition structure generation in multi-agent systems with positive and negative externalities. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 257–263.

Rahwan, T., Ramchurn, S., Jennings, N., and Giovannucci, A. (2009b). An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, 34:521–567.

Rahwan, T., Ramchurn, S. D., Dang, V., and Jennings, N. R. (2007a). Near-optimal anytime coalition structure generation. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2365–2371.

Rahwan, T., Ramchurn, S. D., Dang, V. D., Giovannucci, A., and Jennings, N. R. (2007b). Anytime optimal coalition structure generation. In *Proceedings of the 22nd Conference on Artificial Intelligence*, pages 1184–1190.

Rapoport, A. (1970). *N-Person Game Theory*. University of Michigan Press, University of Michigan.

Resnick, E., Bachrach, Y., Meir, R., and Rosenschein, J. S. (2009). The cost of stability in network flow games. In *Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science 2009*, pages 636–650, Berlin, Heidelberg. Springer-Verlag.

Sandholm, T., Larson, K., Anderson, M., Shehory, O., and Tohmé, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238.

Sen, S. and Dutta, P. S. (2000). Searching for optimal coalition structures. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, page 287, Washington, DC, USA. IEEE Computer Society.

Service, T. C. (2009). Theory and algorithms for coaltion formation among heterogeneous agents. Technical Report HMT-09-01, Human-Machine Teaming Laboratory, Electrical Engineering and Computer Science Department, Vanderbilt University.

Service, T. C. and Adams, J. A. (2010a). Anytime dynamic programming for coalition structure generation. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1411–1412.

Service, T. C. and Adams, J. A. (2010b). Approximate coalition structure generation. In *Proceedings of the 9th International Conference on Artificial Intelligence*.

Service, T. C. and Adams, J. A. (2011a). Coalition formation for task allocation: theory and algorithms. *Autonomous Agents and Multi-Agent Systems*, 22(2):225–248.

Service, T. C. and Adams, J. A. (2011b). Constant factor approximation algorithms for coalition structure generation. *Autonomous Agents and Multi-Agent Systems*, 23(1):1–17.

Service, T. C. and Adams, J. A. (2011c). Randomized coalition structure generation. *Artificial Intelligence*, 175(1617):2061 – 2074.

Shehory, O. and Kraus, S. (1995). Task allocation via coalition formation among autonomous agents. In *Proceedings of International Joint Conference On Artificial Intelligence*, pages 655–661.

Shehory, O. and Kraus, S. (1996). Formation of overlapping coalitions for precedence ordered task-execution among autonomous agents. In *Proceedings of International Conference On MultiAgent Systems*, pages 330–337.

Shehory, O. and Kraus, S. (1998). Methods for Task Allocation via Agent Coalition Formation. *Artificial Intelligence*, 101(1-2):165–200.

Shoham, Y. and Brown-Leyton, K. (2009). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

Shoham, Y. and Leyton-Brown, K. (2009). *Multiaget Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, Cambridge.

Shrot, T., Aumann, Y., and Kraus, S. (2010). On agent types in coalition formation problems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pages 757–764.

Tosic, P. and Agha, G. (2005). Maximal clique based distributed coalition formation for task allocation in large-scale multi-agent systems. In *Proceedings of Massively Multi-Agent Systems I (LNCS)*, pages 104–120.

Ueda, S., Kitaki, M., Iwasaki, A., and Yokoo, M. (2011a). Concise characteristic function representations in coalitional games based on agent types. In *The 22nd International Joint Conference on Artificial Intelligence*, pages 393–399.

Ueda, S., Kitaki, M., Iwasaki, A., and Yokoo, M. (2011b). Concise characteristic function representations in coalitional games based on agent types. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 1271–1272.

Vig, L. (2006). *Multi-Robot Coalition Formation*. Ph.D. dissertation, Vanderbilt University.

Vig, L. and Adams, J. A. (2005). Issues in multi-robot coalition formation. In *Proceedings of Multi-Robot Systems. From Swarms to Intelligent Automata*, volume 3, pages 15–26.

Vig, L. and Adams, J. A. (2006). Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22(4):637–649.

Vig, L. and Adams, J. A. (2007). Coalition Formation: From Software Agents to Robots. *Journal of Intelligent Robotics Systems*, 50(1):85–118.

Voice, T., Polukarov, M., and Jennings, N. R. (2011). Graph coalition structure generation. *CoRR*, abs/1102.1747.

Weiss, G., editor (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press.

Wooldridge, M. and Dunne, P. E. (2004). On the computational complexity of qualitative coalitional games. *Artificial Intelligence*, 158(1):27–73.

Wooldridge, M. and Dunne, P. E. (2006). On the computational complexity of coalitional resource games. *Artificial Intelligence*, 170(10):835–871.

Yao, A. (1977). Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 222–227.

Yeh, Y. (1986). A Dynamic Programming Approach to the Complete Set Partitioning Problem. *BIT Numerical Mathematics*, 26(4):467–474.

Zuckerman, M., Faliszewski, P., Bachrach, Y., and Elkind, E. (2008). Manipulating the quota in weighted voting games. In *Proceedings of the 23rd national conference on Artificial intelligence*, pages 215–220. AAAI Press.